CHIP
CHIP tár
CHIP SPECIAL
14.
Ára: 2350 Ft

LINUX 2.2

Debian GNU/Linux 2.1 Slink

magyar nyelvű telepítővel
és átfogó telepítési útmutatóval

GNOME 1.0

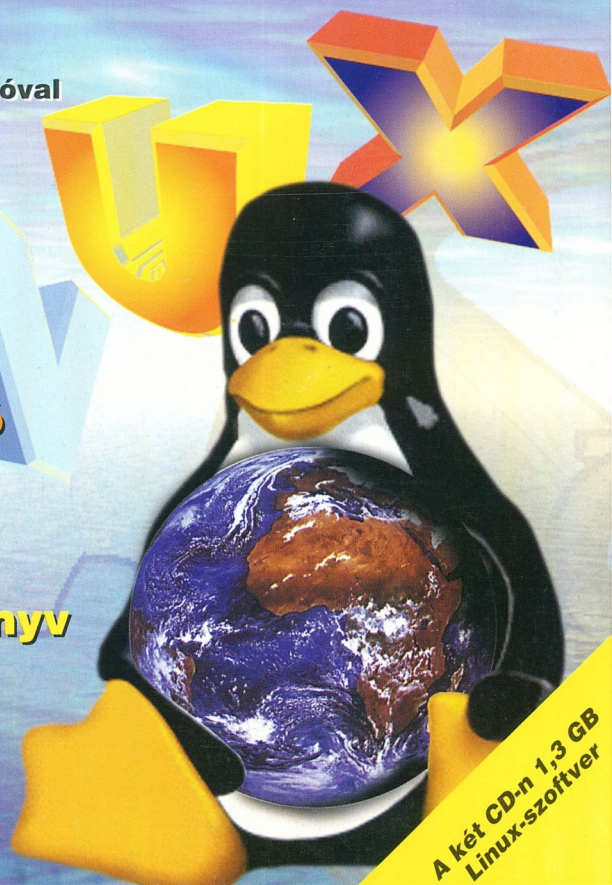
KDE 1.1

Részletes Kernel tanácsadó

Internet PPP-vel

Qmail szakácskönyv

Apache 1.3



A két CD-n 1,3 GB Linux-szoftver

99014

A VOGEL PUBLISHING KFT. LAPJAI ÁTFOGJÁK A VILÁGOT

Media made by Vogel

Vogel Publishing Kft., 1139 Budapest, Hajdú u. 42–44., tel./fax: 350-8731

Beindul a gőzhenger

A CHIP Magazinban 1994 elején, több mint öt éve írtunk először a Linuxról, noha akkor sem volt már újdonság. Az elmúlt öt év szinte síri csöndben telt el a magyar sajtóban, különösebb hír nélkül. Mi azért folyamatosan tudósítottunk e platformról.

Sokan mosolyogtak rajtunk, hogy amolyan kelet-európai operációs rendszert próbálunk a magyaroknak adni. Pedig nem az olcsóság volt a fő cél, hanem az, hogy egy valódi 32 bites, modern, stabil, amolyan „golyóálló" rendszerrel ismertessük meg Olvasóinkat.

Miközben a Windows-változatok verziószámai több mint négyszeresre ugrottak – NT-ből nem volt 1.0, mindjárt hármassal kezdődött –, addig a Linux „csak" most jutott el a 2.2-es kernel verzióig. Kevesen tudják, hogy ez a Unix világ századonként, néhol ezredenként fejlődő verziótörténetében a kőbaltától a fotonhajtóműig való ugrást jelentette, kevesebb mint egy évtized alatt. Miközben a konkurencia a sajtó hangos tapsa közben elfoglalta az asztali gépek piacát, a Linux körbebástyázta magát az Interneten, sőt igazából azt mondhatjuk, hogy az Internet a Linux, és fordítva.

Az elmúlt években a mosoly volt a legtöbb, amivel a nagy számítástechnikai cégek illették a Linuxot.

Sok helyen még kiejteni sem volt szabad e nevet, nehogy a legfőbb partner, a Microsoft megsértődjön.

Az a Microsoft, aki kemény hadjáratot indított a szerver nagyhatalom, a Novell ellen.

A Novell–Microsoft háborúból úgy tűnik mindenki győztesen került ki, mégis a Linux foglalta el nevető harmadikként a fél világot, a Világhálót, ezzel a szerverpiacot is az NT előtt.

Alig több mint fél éve Steve Ballmer, a Microsoft elnöke már a Linuxot nevezte meg az első számú közellenségnek a Novell helyett. Mint a mesében, a megnevezhetetlen szörny megjelent, miután kimondták a nevét.

A sok-sok frusztrált windowsos fejlesztő kapkodó tempóban vetette rá magát az új, már „hivatalosan" is elismert piacra. A Corel, az IBM, a Compaq, az Adobe, az Oracle és sok más kisebb-nagyobb cég jelentette meg, illetve jelentett be küszöbön álló linuxos termékeit.

A sajtóval is hasonló a helyzet: egyre több cikk jelenik meg a Linuxról olyan lapokban, ahol korábban szintén csak a Windows volt az egyetlen megnevezhető operációs rendszer.

Lelkendező cikkek születnek olyan emberek tollából, akik még a Windowst sem tudják egyedül telepíteni. A Linux immár pénzt jelent, a pénz sajtót jelent.

Nagy ereje, robosztus hatékonysága ellenére a Linux még mindig nem kezes bárány. Nehéz telepíteni, a rendszert konfigurálni, és általában minden új.

Egyre jobb grafikus kezelőfelületek jelennek meg – CD-nk Linuxával a GNOME, a GNU hatékony, Windowszal vetekedő héja jár –, ám mire eljutunk a grafikus felület indulásáig, gyakran visszasírjuk a Windowst.

Kiadványunk legfőbb célja a nehézségek átlépése. CD-inkre a Debian GNU Linux került fel, ez egyben az egyik leghatékonyabb, ámde az egyik legnehezebben konfigurálható Linux rendszer. Szervernek és munkaállomásnak, vagy a kettőnek egyben is telepíthető. Erről szól CHIPtárunk, mit hogyan tegyünk fel és konfiguráljunk, ha webszervernek, vagy „csak" otthoni gépnek szeretnénk Linuxot.

Magyarul, magyaroknak! Hála a forráskód szabadságának, részben magyarítani tudtuk a rendszert.

Mind a telepítő, mind a GNOME magyarul „beszél", így a kezdet könnyebb lesz. A folytatást is igyekszünk segíteni, e CHIPtárunkban és a későbbi CHIP Magazinokban tovább támogatjuk az érdeklődőket, mint ahogy azt már öt éve tesszük.

Elindult a gőzhenger, azt ist tudjuk merre tart.

Tartsanak velünk!

Tartalomjegyzék

Linux 2.2

CHIPTÁR: Linux 2.2

Felelős kiadó: **Ivanov Péter** ● Szerkesztők: **Czakó Krisztián és Kerecsen Tamás** ● Olvasószerkesztő: **Nagy Anna** ● Címlapgrafika: **Stefan** ● Korrektor: **Sartoris Adrienn és Felvégi Emese** ● Kiadja: Vogel Publishing Kft. 1139 Budapest, Hajdú u. 42–44. ● Terjeszti: Hírlapkereskedelmi Rt. és a Nemzeti Hírlapkereskedelmi Rt. regionális részvénytársaságai. ● Megrendelhető telefonon: (36-1) 349-4768-as vagy telefaxon a (36-1) 350-8731-es számon ● Montírozás és nyomás: a Tipász stúdió gondozásában készült. Felelős vezető: Nagy Lajos

HU ISSN 1219—4522

Milus János
johans@lme.linux.hu

Linuxosok egyesülete

1998. szeptember 27. pirosbetűs ünnep a hazai linuxosok számára. Ekkor alakult meg az első országos szervezet, a Linux-felhasználók Magyarországi Egyesülete – röviden LME.

Megalakult – de jogilag sajnos még nem jött létre, legalábbis amikor e sorokat írom.
Nem számítottunk olyan hosszú és fáradságos jogi procedúrára, mint amin eddig már végigmentünk, és ami feltehetőleg még előttünk áll. Ha több, váratlan akadály nem jön közbe, az egyesület várhatóan április végére lesz bejegyzett, bankszámla- és adószámmal rendelkező szervezet, amelyet néhány év múlva akár az adója 1%-ával is támogathat bárki.

Célok

Mi is a célja az LME-nek? Manapság mindenki tapasztalhatja, hogy megfelelő marketing/PR munkával gyakorlatilag bármit el lehet adni, és ez ma nagyon hiányzik a magyarországi Linux-kultúrából. Hiszen a Linux hiába a leggyorsabb és legstabilabb operációs rendszer a PC-kategóriájú szerverek piacán, a döntéshozó vezetőkig gyakran úgy jut el a híre, mintha csupán néhány fiatal játékszere lenne. Ezen a helyzeten minél előbb változtatni szeretnénk.

Annak érdekében, hogy a szakmai berkekben meg tudjuk ismertetni a Linux valódi értékeit és az LME működését, részt kívánunk venni a jelentősebb számítástechnikai kiállításokon. Ezen túlmenően az egyetemistáknak – mint leendő döntéshozóknak – és a különböző vállalatok vezetőinek roadshow-kat szervezünk.

Tekintve, hogy a Linux részesedése meghaladja a 20 százalékot az operációs rendszerek piacán, remélhetőleg Magyarországon is hamar elindul az a folyamat, ami világszerte megfigyelhető. Olyan nagy nevek állnak a Linux-mozgalmak mögé, mint az IBM, a Compaq, a SUN, az Oracle, az Informix, az Intel, a Corel, és még sorolhatnánk. A változás – a hazai piacot ismerve – először valószínűleg a hardverforgalmazók körében következik be, hiszen egy ingyenes szoftver sokkal olcsóbb, mint egy teljes árú vagy OEM termék. Nekik egy külön programot szeretnénk indítani, melyben az egyesület segítségével megszerezhetik a pingvin logót.

Aki ilyen boltban vásárol, biztos lehet abban, ha megkérdezi, hogy az adott hardvernek milyen a linuxos támogatása, az eladó nem hiszi azt, hogy egy trópusi betegség felől érdeklődik.

A pingvin logót megszerző boltok listáját természetesen rendszeresen közzé fogjuk tenni weboldalunkon.

Oktatás

Jelenleg Magyarországon nincs olyan tananyag, ami alapján el lehetne sajátítani a Linux használatát, adminisztrálását. Ezért javasoljuk a Linux beillesztését a NAT-ba.

Nem alakult még ki világviszonylatban egy az ECDL-hez hasonló vizsgarendszer, ami alapján mérni lehetne valakinek a Linuxban való jártasságát. Ismerve az iskolák anyagi helyzetét és az operációs rendszerek árát, valószínűleg sokan döntenének a Linux használata és oktatása mellett. E helyzet megváltoztatásához próbál segítséget nyújtani az egyesületen belül működő Linux Oktatási Projekt, röviden LOP. Olyan, akár órán, akár szakkörön oktatható magyar nyelvű tananyag létrehozása a cél, amelynek moduláris a felépítése. Ezáltal bármilyen igényt kielégíthet az egyszerű irodai programcsomagok használatától kezdve, a professzionális rendszergazda-képzésig. Ha elkészül az anyag, szeretnénk megnyerni hozzá az Oktatási és Közművelődési Minisztérium támogatását is.

Az előttünk álló út nem teljesen járatlan, hiszen világviszonylatban már nem példa nélkül való a Linux használata és oktatása a középiskolákban.

Gondoljunk itt például Mexikóra, ahol a magyar SuliNet programhoz hasonlóan több száz középiskola kapott számítógépeket és Internet-elérést. A mexikói programban a gépek operációs rendszere a Linux volt. Valószínűleg, ha a

SuliNet program keretében linuxos gépeket kaptak volna az iskolák, a program ára 15–20 százalékkal csökkenthető lett volna. (Tapasztalatunk szerint, a SuliNet projektben géphez jutott intézmények közül igen sokban a számítógéphez kapott rendszer helyett a Linuxot választották így is. – a szerk.)

A számítástechnika manapság már kikerült az elefántcsont-toronyból, így nem lehet arra számítani – ha a desktop-felhasználást vesszük célba –, hogy a felhasználó beszél angolul.

Jelenleg nincs magyar nyelvű Linux, és az itthoni elterjedésének ez nagymértékben gátat szab. (Ma már elég jól áll a fordítás, a CD-nken közreadott Debian telepítője és a dokumentáció egy része magyar, valamint a S.u.S.E Linux telepítője és konfigurációs programja is magyar már. – a szerk.) Ezért dolgozik az LME berkein belül egy, a magyarítással foglalkozó munkacsoport.

Reményeink szerint az elkészült fordítások bekerülnek a népszerűbb disztribúciókba, de terveink között szerepel egy minden ízében magyar nyelvű disztribúció létrehozása is.

Az eddig elkészült magyarításokat le lehet tölteni weblapunkról. Persze az LME tevékenysége nem merül ki ennyiben. Lehetőségeinkhez mérten szeretnénk szakmai fórumot biztosítani a felmerülő technikai kérdéseknek, illetve a Linuxhoz kapcsolódó ötleteknek, kezdeményezéseknek.

Tagjainknak ingyenes e-mail postafiókot biztosítunk, és rendezvényeinken is kedvezményesen vehetnek részt. (Az egyesület által időközben elindított rendezvények egyike egy ingyenes, heti egy alkalommal tartott oktatás, melyről a http://lme.linux.hu/oktatas/ weblapon olvashatók további információk. – a szerk.)

Ha többet akar rólunk megtudni, keresse fel a www.linux.hu weblapot, illetve keressen meg minket személyesen az Info '99 kiállításon, ahol a Vogel Publishing Kft. – a CHIP Magazin és a CHIPtár sorozat kiadója – biztosít számunkra megjelenési lehetőséget, melyért ezúton is szeretnénk köszönetet mondani. ■

Horváthné Harmati Szilvia -(hszilvia@kvark.szif.hu) Horváth András (horvath@rs1.szif.hu)

A Linux rövid története

Egyre többet hallani ma a sajtóban a Linuxról, egyre többet emlegetik, hogy mit tud, mennyire stabil és gyors rendszerré vált a sokéves internetes fejlesztés eredményeképpen, de ritkán olvashatunk arról, hogyan is indult, keveset hallunk a történetéről. Mi itt most a kernel (a rendszer magja) fejlődését követjük nyomon a 2.0-ás verzióig. A 2.2-es, azaz napjaink legfrissebb Linuxáról egy külön cikkben írunk.

A Linux születése, csecsemőkora (0.01–0.10)

A Linux története úgy kezdődött, hogy egy finn egyetemista – Linus Torvalds –, megunta az akkoriban oktatási célokra olcsón beszerezhető PC-s Unix, a Minix kötöttségeit, hibáit, és elhatározta, hogy megpróbál egy olyan rendszert összehozni, ami saját magának egy jobb környezetet biztosít.

Legelőször a 80386 processzor védett módú (protected mode), feladat-váltó (task-switching) lehetőségeivel szeretett volna megismerkedni. Ez kb. 1991 nyarának elején lehetett. A pontos dátumra maga a szerző sem emlékszik, de amikor egyszer megkérdezték, mikor van a Linux születésnapja, azt válaszolta: nem tudja megmondani, de egy e-mail tanúsága szerint július 3-án már a POSIX szabvány után érdeklődött, így akkor már biztos futott az alaprendszer.

A program fejlesztése Minix alatt történt, eleinte assemblyben. Az első fázisban kialakuló 0.01-es változat még semmire sem volt használható, csak egy lépcső volt a továbbfejlesztéshez. Linus Torvalds így ír erről:

„Két hónap telt el az alaprendszer felállításáig, de utána rövidesen lett egy lemezmeghajtóm (amely súlyos hibákkal volt tele, de az én gépemen történetesen működött) és egy kis file-rendszerem. Körülbelül ekkor – 1991. augusztusának végén – tettem közzé a 0.01-es változatot: nem volt kicsinosítva, nem volt floppymeghajtója, és nem sok mindent tudott csinálni. Azt hiszem, soha senki nem fordította le ezt a változatot. De akkor már beindultam, és nem akartam addig megállni, amíg túl nem haladom a Minixet."

Amikor Linus áttért a C nyelvre, a fejlesztés lényegesen gyorsabbá vált, és olyan nagyratörő tervek fogalmazódtak meg, hogy valaha le lehessen fordítani a GNU C fordítóját Linux alatt. (Ma már csak csodálkozni lehet azon, hogy 1991-ben ez volt a nagy álom, és azóta hol tart a rendszer.)

Ez a legelső változat még nem volt használható: csak Minix alatt lehetett lefordítani, és semmi hasznos funkciója nem volt azon kívül, hogy írója megismerkedett a processzorral.

1991. október 5-én hirdette meg Linus az első „hivatalos", 0.02-es Linuxot az Interneten. Ekkor már néhány alapprogram futott a rendszeren (pl.: a GNU gcc nevű C fordítója, valamint a bash burokprogram), így már el lehetett kezdeni használni a rendszert. Ekkor nem is a rendszer használhatóságának növelése volt a fő cél, hanem a rendszermag fejlesztése. Ezért ekkor nem készültek dokumentációk, installációs csomagok stb. A Linux ekkor még csak a megszállott hackereknek készült.

Linus ekkor elhatározta, hogy az Interneten keresztül bevonja a fejlesztésbe a szabad kapacitással rendelkező programozókat, és a következő hirdetményt tette közzé a comp.os.minix hírcsoportban:

„Sóvárogsz a Minix-1.1 szép napjai után, amikor a férfiak igazi férfiak voltak, és mindenki maga írta a saját eszközmeghajtóját? Egy szép project nélkül vagy, és épp fened a fogad egy operációs rendszerre, amit igényeidnek megfelelően alakíthatsz? Frusztrálónak találod, ha minden működik Minix alatt? Akkor ez a levél lehet hogy pont neked szól.

Ahogy egy hónapja említettem, egy szabad Minix-szerűségen dolgozom AT-386 számítógépre. Végül is elérkeztem egy olyan állapotra, amikor ez egyátalán használható (bár ez függhet attól, mit akarsz), és a program forráskódját szélesebb körben tervezem szétosztani. Ez még csak a 0.02-es változat, de sikeresen futtattam a bash, gcc, gnu-make, gnu-sed, compress stb. programokat alatta."

Megjegyzendő, hogy ekkor, és még egy darabig a Linux erősen kapcsolódott a Minixhez: önállóan nem is létezett, csak alatta lehetett lefordítani, futtatni, továbbá az Interneten is a Minix hírcsoportjában folyt a Linux fejlesztésével kapcsolatos információcsere.

A 0.03-as verzió két-három hét alatt megszületett, majd 1991. decemberében Linus kibocsátotta a 0.10-eset is. Ez az ugrás a számozásban azt tükrözte, hogy jelentősen megnőtt a Linux alatt futtatható alkalmazások száma, de a Linux még mindig nem volt önálló, szerzője szerint is „egy hacker által hackereknek írt" rendszerről van szó, így a rendszernek csak fejlesztői vannak, felhaszálói nem.

A Linux gyermekkora (0.11–0.99)

1991. december 19-től, a 0.11-es változat kibocsátásától számíthatjuk a Linux gyermekkorát. Ez volt az első önálló rendszer, tehát nem kellett Minix a használatához. Sok olyan tulajdonsággal rendelkezett, amely jelezte, hogy itt valami komoly készül. Ezeket Linus felsorolásában adjuk közre:

„A 0.11-nek a következő újdonságai vannak:
demand loading;
kód és adatmegosztás nem kapcsolódó processzek közt;
sokkal jobb floppyvezérlők (most már többnyire működnek);
hibajavítások;
Hercules/MDA/CGA/EGA/VGA támogatás;
a konzol hangot is ad (Óh! Fantasztikus rendszermag!);
mkfs/fsck/fdisk;
amerikai/német/francia/finn billentyűzet;
a com1/2 sebessége beállítható."

A 0.12-es változat 1992. január 15-én látott napvilágot, néhány bővítéssel: Már volt init/login szolgáltatás (nem rootként kellett először bejelentkezni, és inicializálni a rendszert), közelített a POSIX szabványhoz, virtuális memóriát is használt és kisebb korrekciókat tartalmazott.

Ez már egy elég stabil változat lett, ekkortól kezdődött el a Linux igazi hódítása. A 0.12-es Linuxot ugyanis elkezdték használni „egyszerű" felhasználók is, nemcsak megszállottak.

Szintén ehhez a változathoz kapcsolódik a Linux fejlesztésének kiterjesztése: a 0.12-es már lényeges részeket tartalmazott, melyeket nem Linus Torvalds írt. Pl.: a job controlt Theodore Ts'o, a virtuális konzolokat Peter MacDonald programozta.

Az így előálló rendszer már több vonatkozásban jobb volt a Minixnél, de még mindig nem volt látható, hogy ebből akkora mozgalom lesz, mint ami mára kialakult.

A sikeren felbuzdulva a verziószám hirtelen ugrott: a 0.95-ös 1992. márciusában, a 0.96 áprilisban lett kibocsátva. Ettől a pillanattól kezdve hihetetlen ütemben gyarapodott a Linux-felhasználók és -programozók száma.

Ekkor a verziószám hirtelen „befékezett": 1993. decemberében a verziószám 0.99pl14 volt. (A pl14 a *patch level 14* rövidítése, azaz ez a 14. javított változat.) Bár a 0.95-ös verziótól kezdve a szolgáltatások száma, a megbízhatóság, és más egyéb szempontból jelentős javulás következett be, ráadásul hihetetlenül sokan használták ezeket a rendszermagokat, az 1.0 verziószámot mégis csak akkor merték kiadni (1994. elején), amikor a POSIX szabvánnyal való kompatibilitás kielégítővé vált.

A 0.95–0.99 rendszermagra épülő rendszereknek óriási népszerűségük volt. Egyes egyetemeken, pontosabban azok bizonyos intézeteiben gyakorlatilag likvidálták az összes nem linuxos programot, és a PC-ken nem lehetett DOS-t vagy Windowst találni. (Legfeljebb a titkárságokon.) Ez főleg olyan helyeken következett be, ahol a kutatók UNIX alatt dolgoztak, mert egy linuxos PC-n otthon is fejleszthették a programjukat, és ezeket egyszerű volt az intézet nagykapacitású gépeire áttenni.

(Egyikünk személyesen tapasztalta ezt 1993-ban a Würzburgi Egyetem Csillagászati Tanszékén: szó szerint senki sem használt DOS-t, még otthon sem, legfeljebb a kedvenc játékprogramjának futtatására. Az otthoni linuxos gépen kifejlesztett, tesztelt programok gond nélkül áttehetők voltak az intézet CRAY gépére.

Hazánkban ezidőtájt (1993.) kezdett igazán elterjedni a Linux, mert ekkorra kötötték be a felsőoktatási intézmények nagy részét az Internetre, így sokaknak megnyílt a lehetősége a Linux beszerzésére.

A Linux és a Minix szétválását meggyorsította, hogy a Minix szerzője, Andrew Tanenbaum nem nézte jó szemmel a Linuxot. Alapvetően elhibázottnak tartotta a Linux rendszermag típusát; Linus Torvalds ugyanis ún. „monolitikus kernelt" írt, míg Andrew Tannenbaum (elméleti megfontolások alapján) a

mikrokernelt jobb választásnak tartotta. Sajnos, a vitába személyes elemek is keveredtek, és a vita az Internet nyilvánossga előtt zajlott, így kissé ideges hangulatban zajlott le a Minix és a Linux szétválása. Csak ízelítőül:

Andrew Tannenbaum: „Továbbra is fenntartom azt, hogy 1991-ben monolitikus kernelt tervezni alapvető hiba. Örülj, hogy nem vagy a tanítványom. Nem kapnál túl jó minősítést egy ilyen tervezésre :-)"

Linus Torvalds: „Az Ön foglalkozása professzor és kutató: Ez egy pokolian jó mentség a Minix némely agysérülésére."

Andrew Tannenbaum (1992-ben): „A Linux elavult!"

Mivel nem ismerem a Minixet, ezért nem tudok döntőbíró lenni a vitában, de mára a Linux felhasználók száma, az alkalmazások sokrétűsége, és egy csomó más szempont szerint nagyságrendekkel veri a Minixet. Talán elég azt megjegyezni, hogy a Linuxszal kapcsolatos angol nyelvű hírcsoportok száma több, mint 10, és magyarul is jó néhány (kb. 6) linuxos hírcsoport működik, míg a Minixszel csak egy foglalkozik az egész világon, melynek forgalma össze sem vethető még a magyar Linux-listákéval sem.

Annyit azonban meg kell jegyezni, hogy a Minix azért sem terjedhetett annyira, mint a Linux, mert nem szabad terjesztésű. Ennek ellenére az valószínűsíthető, hogy Andrew Tanenbaumnak nem volt igaza abban, hogy a Linux alapvető szervezése teljesen hibás, mert alapvető hibákkal nem lehetne egy rendszer ennyire stabil, és nem is terjedt volna el. Talán személyes ambícióit sértette, hogy egy egyetemista az övénél használhatóbb rendszert hozott össze, mely mellett a Minix elhanyagolható szerepet játszik. Mindenesetre sajnálatos, hogy ilyen rivalizálásra is sor kerülhet komoly szakemberek között.

A Linux fiatalkora (1.0.0–1.2.13)

A POSIX szabványosítás megfelelő szintű elérésével 1994. márciusában megjelent az 1.0.0 sorszámú kernel. Ekkortól kezdve egy speciális sorszámozási eljárást vezettek be a fejlesztők: A verziószámot három, ponttal elválasztott nemnegatív egész jelzi. Az első a fő verziószám, ami csak a rendszermag lényegét érintő változásoknál vált eggyel nagyobbra. A második szám elég speciális jelentésű: ha páros, akkor stabil, tesztelt kernelről van szó, amit bárkinek ajánlanak használatra, míg a páratlan szám tesztváltozatot jelöl, amit inkább azoknak javasolnak, akik tesztelni, fejleszteni szeretnék a kernelt, és akiknek nem számít, ha a rendszer néha elszáll. A harmadik szám pedig kisebb módosításokkor ugrik egyet.

Ennek megfelelően egyszerre két szálon fut a legújabb verziószám: pl. e cikk írásakor a legfrissebb két kernel sorszáma 2.0.30 illetve 2.1.46. A stabil verziókba csak olyan modulok kerülhetnek bele, amelyek a fejlesztői változatokban már üzembiztosnak bizonyultak. Ez a fura sorszámozás lehetővé teszi, hogy az egyszerű felhasználók csak a valóban használható változatokat kapják meg, de közben az esetleg még hibákat tartalmazó fejlesztői változatok is hozzáférhetőek legyenek. A későbbiekben mi csak a stabil verziószámokkal foglalkozunk, mert ezek a „hivatalos" változatok.

Ezen változatok nem hoztak újabb hatalmas áttörést, mert az már korábban bekövetkezett. A fejlesztés során a rendszermag egyre hatékonyabb lett, beépítették a legújabb hardverek meghajtóit (CD-olvasók, PCI-buszok, újabb SCSI-eszközök stb.). Talán leginkább az 1.2.x-es kernelekkel bevezetett új végrehajtható programformátum, az ELF megjelenését kell itt megemlíteni.

Ebben az időben a Linux alatti felhasználói programok száma nőtt meg hihetetlenül. Míg korábban főleg már meglevő szoftvereket vittek át a Linux alá, ezután már megjelentek azok a programok, melyeket Linux alatt fejlesztettek, és innen vittek át a többi rendszerre. Ekkora már nagy szoftvercégek is elkészítették programjaik linuxos változatát (pl.: Maple V, Motif 2.0). Ezek természetesen nem szabadterjeszésűek, és az, hogy megéri Linuxra is adaptálni őket, egyértelműen jelzi, hogy a Linux-felhasználók tábora világméretekben is piacot jelent.

Ekkortájt egyre több Linux-disztribúció kezd megjelenni, azaz több cég olyan programcsomagot állít elő, amelyekkel a Linux telepítése, karbantartása sokkal könnyebb, mintha mindenki egyenként gyűjtené be a rendszer részeit.

A törekvő Linux (2.0.0–2.0.36)

1996. augusztusában jelent meg a 2.0.0 sorszámú rendszermag. Ennek fő újítása a modulok megjelenése volt: a kernel bizonyos részei modulként is elkészíthetők, és ezek a modulok akár automatikusan, akár kézzel betölthetők a memóriába, ahonnét a rendszer kipakolja őket, ha régóta nem használjuk.

Például a nyomtató, floppyvezérlő, nem linuxos file-rendszereket kezelő részeket célszerű modulba tenni, mert ekkor ezek csak addig foglalják a memóriát, amíg éppen használjuk őket, és ez többnyire a

munkaidő jelentéktelen része. Ezzel az az érdekes helyzet állt elő, hogy a rendszermag memóriaigénye *kisebb* lett, míg hatékonysága és megbízhatósága megnőtt. (Erről a tendenciáról példát vehetnének a piacot uraló szoftverhatalmak is. Csakhogy azok nem a hatékonyabb, hanem a nagyobb anyagi hasznot hozó rendszer írásában érdekeltek, és a hardvergyártókkal való összefonódás miatt sokszor a nagyobb memóriaigény kifejezetten előnyös a cég számára.)

Az is megemlítendő, hogy ekkorra már mintegy 200 hacker kódja került be a stabil kernelbe, és Linus Torvalds idejének jelentős részét ezen munka koordinálása köti le. Az, hogy ilyen szétszórt fejlesztés ellenére a rendszer stabil, működőképes, legalább akkora érdeme Linusnak, mint az, hogy elindította az egész fejlesztést.

A Linux és a GNU kapcsolata

Fontos, hogy szóljunk pár szót a Linux és a GNU kapcsolatáról is, hiszen ez döntő volt a Linux fejlődése szempontjából:

Egyrészt, a Linux rendszermag (az első néhány verzió kivételével) a GPL (GNU Public License) hatálya alá esik, másrészt a Linux rendszerek alapprogramjai és a felhasználói programok jelentős hányada a GNU project keretében készült, vagy írója a GPL-t alkalmazza.

Bizonyos szempontból a Linux kiegészíti a GNU projectet. Mint olvashattuk, a GNU egy teljes rendszer szeretne lenni, de ezidáig még nem írtak rá megbízható rendszermagot. (Jelenleg a GNU rendszermag, a Hurd, a 0.2-es verziónál tart, de ez még nem teljesen stabil változat.) Linus Torvalds és társai viszont pont ezt tették meg, igaz, nem a GNU keretein belül.

Nem látható pontosan előre, hogy hogyan alakul a jövő. Az is lehet, hogy a GNU rendszermag nem fog elkészülni, vagy legalábbis elterjedni, mert a Linuxot nehéz lesz túlszárnyalni. De az sem kizárható, hogy Richard Stallman egyszer csak mégis előáll a GNU kernellel, ami lesöpri a Linuxot a színről. (Bár ezt kevéssé tartom valószínűnek.)

A Linux jövője

Nehéz, és veszélyes dolog jóslásokba bocsátkozni, de néhány alapvető dologban biztosak lehetünk:

A Linuxnak mára akkora tábora van, hogy még Linus Torvalds és a többi vezető programozó esetleges kilépése esetén is tovább fog folytatódni a munka. Különösen igaz ez azért, mert mára nemcsak IBM PC-ken, hanem a legterjedtebb munkaállomásokon is fut Linux, és ezen Linux-változatok némelyikét nem is Linus Torvalds koordinálja.

A mai napig folyamatosan növekszik a linuxos szoftverek, a felhasználók száma, és az alkalmazások sokrétűsége. Hogy csak egy példát említsek: egyetemeken és kutatóintézetekben egyre inkább terjed az a szokás, hogy linuxos PC-ket hálózatba kötnek, és az így előálló hardvert valamilyen párhuzamos programfejlesztési környezet alól használják. Az ilyen rendszereknek sokkal kedvezőbb az ár/teljesítmény aránya, mint a készen kapható kisebb szuperszámítógépeké.

Úgy néz ki tehát, hogy lendületben van a rendszer, és több évig tartó biztos dinamikus fejlődés előtt áll.

Véleményem szerint az egyetemi és kutatási helyeken méltó vetélytársa bármelyik úgynevezett „operációs rendszernek", és itt a jelenleginél sokkal nagyobb elterjedésre számíthatunk a közeljövőben. (Különösen, ha figyelembe vesszük a magyar felsőoktatási intézmények anyagi helyeztét.)

A rendszer stabilitása miatt már most is sok helyen alkalmazzák hálózati szervernek a linuxos gépeket, de még itt is nagy lehetőségek rejlenek a Linux terjedése előtt.

Az biztos, hogy Linus Torvalds sem lesz a fejlődés gátja. Egy helyen ugyanis ezt írja:

„Azon a napon, amikor a Linuxot valaki más jobban tudja szolgálni (az FSF egy természetes alternatíva), félreállok. Nem gondolom, hogy emiatt valakinek aggódnia kellene, és nem is gondolom, hogy ez a közeljövőben bekövetkezik. Szeretek a Linuxszal foglalkozni, még ha ez egy kis munkával jár is, és még senkitől sem kaptam panaszt. (Csak néhány, majdhogynem szégyenlős emlékeztetőt arról, hogy egy patchet elfelejtettem, vagy nem vettem figyelembe, de eddig semmi negatívat.)

Ne vegyétek úgy a fentieket, hogy azon a napon, amikor valaki ellenkezik, abbahagyom: elég keményfejű vagyok – Lasu, aki ezt a vállam felett olvassa, megjegyezte, hogy a fafejű közelebb van az igazsághoz – egy kis mocskolódáshoz. Ha nem lennék az, abbahagytam volna a Linux fejlesztését, amikor a comp.os.minixen nevetségessé tettek. Csak arra gondoltam, hogy bár a Linux eddig az én gyermekem volt, nem akarok útban lenni, ha az emberek jobbat szeretnének csinálni (*).

(*) Hé, lehet, hogy kérhetném a szentté avatásomat a Pápától! Tudja valaki az e-mail címét?"

Slapic
slapic@vogel.hu

A Debian GNU/Linux 2.1 telepítése

Az egyik legnagyobb eltérés a különféle Linux disztribúciók között a telepítőprogram filozófiája és kezelése. Az azonban közös valamennyi telepítőben, hogy igyekeznek minél egyszerűbbé tenni a telepítést.

Ma az egyik fő törekvés, hogy a Linux a professzionális szervervilágból és a „szakemberek társának" pozíciójából kinője magát, és igazi erős desktop operációs rendszerré váljon. Ez a törekvés jellemzi az olyan üzleti disztribúciókat, mint a RedHat, a S.u.S.E, vagy a Caldera. A Debian, amit kiadványunk mellékleteként kezébe kapott az Olvasó, más. Készítője, az SPI, azaz Software in the Public Interest nem üzleti vállalkozás, hanem – hasonlóan a Linux alapelveihez – kizárólag szabad szoftvert készítenek. Céljuk nem a profit, hanem egy jól működő rendszer megalkotása. Mivel ezek az emberek nem marketing oldalról, hanem a szakember oldaláról közelítik meg a problémát, a rendszer kezelése és működése is ehhez igazodik. Tehát senki se várja el a Debian telepítőjétől, hogy 15–20 perc alatt egy működő rendszerre tehet szert. Aki erre vágyik, próbálja ki a CHIP Magazin CD-mellékletén alkalmanként megjelenő RedHat vagy S.u.S.E disztribúciókat.

A Debian esetén a telepítés célja egy kész, működő és beállított rendszer, míg az előbb említetteknél a testre szabás a telepítéstől független, azután következő folyamat.

A Debian-telepítő – illetve az egyedi csomagokat beállító program – másik, elsőre nehézségnek tűnő elve az, hogy azt feltételezi a használójáról, hogy használta már a Linuxot. Ez utóbbi kezeléséhez próbálunk meg segítséget nyújtani. Nem kell megijedni, azaz – Douglas Adamsszel élve – DON'T PANIC! Alaposan el kell olvasni a gép üzeneteit – ahol módunkban állt, ott magyarra fordítottuk –, de még így is jócskán fog találkozni az Olvasó angol nyelvű szöveggel, így jól jön az angol nyelv ismerete is. Annyit azonban megígérhetünk, hogy aki az akadályokon végigküzdötte magát, az a telepítés és a beállítások elvégzése után egy stabil, biztonságos és jól működő rendszer birtokosa lesz. Ennek a jövőbeli frissítése, karbantartása és felügyelete már szinte gyerekjáték.

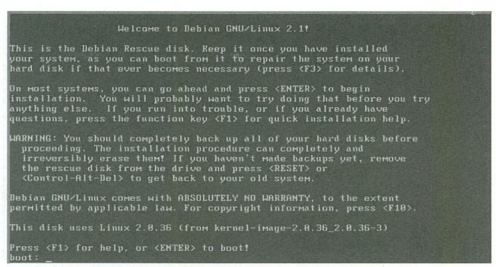
Amit már a telepítés előtt célszerű tudni

A legfontosabb feladat, mielőtt egy adott gépre elkezdenénk bármilyen operációs rendszert telepíteni, a biztonsági másolat elkészítése. Létfontosságú adatainkat mindig mentsük le előzetesen. Még a legképzettebb szakember is elkövethet (az adatokra) végzetes következményekkel járó hibát.

Mennyi hely szükséges, és azt hogyan osszuk be?

Ha még sosem volt Linux a számítógépünkön, biztosan particionálni kell a merevlemezt, ami – ha törlünk is – valószínűleg adatvesztéssel jár együtt.

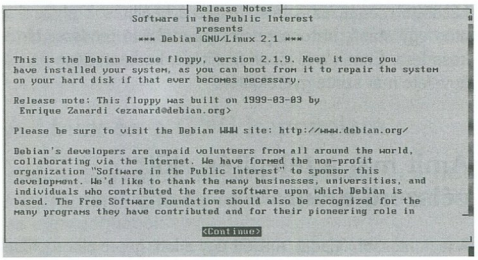
Ha a Debian Linux lesz az egyetlen operációs rendszer a gépen („Gratulálunk, tökéletes választás!"), el lehet kezdeni a telepítést, de még így is javasoljuk az előkészületi lépések gondos végigolvasását. Ha bármilyen más operációs rendszer is van vagy lesz, külön kell biztosítanunk helyet a Linuxunknak. Ebben az esetben kétféle partíciótípust kell majd használnunk:



A Debian telepítő kezdőképernyője

a Linux-natívot és Linux-swapet. Az előbbire a Linux extended 2 file-rendszere, az utóbbira a swap (avagy csereállomány) kerül. Bármelyikből használhatunk egyet vagy többet is. Legalább egy Linux-natív partícióra azonban szükségünk lesz, valamint nagyon ajánlott egy swap-partíció létrehozása is. Az előbbi, ha csak egy partíciót használunk, akkora legyen, hogy elférjen rajta az operációs rendszer és az adataink is. Ha mindent felteszünk a CD-kről, több gigabyte helyre lesz szükség, de ezt semmiképpen sem javasoljuk.

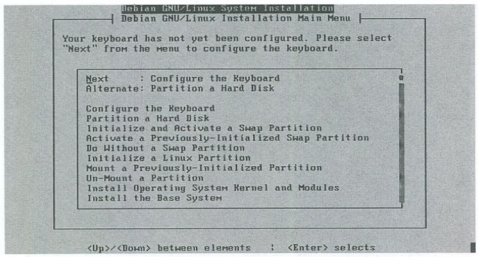
Eleve lehetetlen is minden programot telepíteni, mivel akadnak köztük olyanok, melyek ugyanazt a funkciót látják el. (Sokszor előfordul, hogy maximum egy program láthat el egy adott funkciót. Ilyen például a levelezőszerver.)



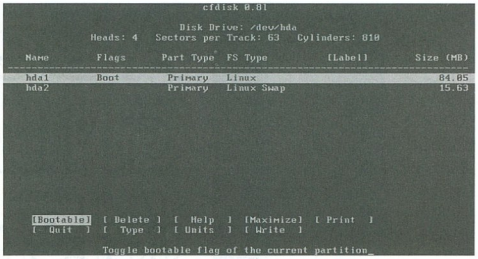
Információk a telepítendő Debian GNU/Linuxról

Egy átlagos rendszer 2–300 MB-ot foglal el. A cikk szerzője például sok programot szeret kipróbálni, de nem szereti utána letörölni azokat, így csak a rendszere közel három gigabyte-on terpeszkedik el, viszont üzemeltet olyan szervert, ahol a rendszer mindössze 100 MB-nyi helyet foglal. A swappartíció mérete maximum 128 MB (pontosabban maximum ennyit fog használni a rendszer, hiába nagyobb), de ebből lehet több is.

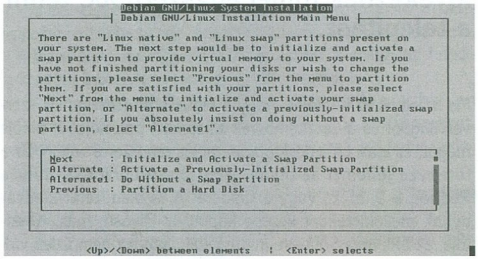
Ajánlott a rendelkezésre álló memória kétszeresét, vagy speciális esetben az igényeknek megfelelő mennyiséget alkalmazni. Ne gondolja senki, hogy olyan sok memóriája van, hogy nem lesz szükség swapre. Szükség lesz rá. Nemcsak a fizikai memória kibővítésére, hanem az ugyan memóriában levő, de nem futó programok tárolására is, azért, hogy az értékes memóriát a rendszer használhassa. Annyit érdemes itt megjegyezni a Linux memóriakezeléséről, hogy igen aktívan használ, illetve kihasznál minden bitnyi memóriát. Ritkán látunk üresen maradt memóriaterületet.



A telepítő főmenüje, mely felajánlja a következő logikus lépést



A cfdisk particionáló program



Ideje inicializálni a swap partíciót

Ez a gyakorlatban annyit jelent, hogy a gyorsítótár (diskcache) mérete dinamikusan változik; ha van rá igény, kitölti a teljes memóriát.

Sőt, mint fentebb említettük, a nem futó programot inkább kiteszi swapre a rendszer, ezáltal növelve a gyorsítótár méretét.

Partíció-csipegetés

Mivel a Linuxnak saját partíciótípusa van, telepítéséhez szabad, nem particionált lemezterületre lesz szükségünk. Ha ez nem áll rendelkezésünkre, valahogy fel kell szabadítani elegendő helyet.

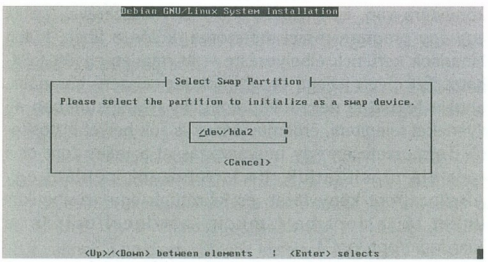
Erre több lehetőségünk is van. Vegyük sorra ezeket!

Ha DOS vagy Windows van a gépünkön, először töredezettség-mentesítsük a lemezt. Ezután az üres részt DOS-módban (és nem DOS-ablakban!) a CD-n megtalálható DOS-os fips programmal levághatjuk a végéből. A Linuxnak mindegy, hogy egy elsődleges (primary) partícióra vagy egy

bővített (extended) partíció logikai egységére kerül. Ezen túlmenően az is igaz, hogy elvileg korlátlan számú elsődleges (primary) partíciót és logikai egységet képes kezelni, de ezt a PC-s rendszer elvei korlátozzák: maximum négy elsődleges, vagy három elsődleges és egy bővített partícióra. Ez utóbbin belül már kedvünkre hozhatjuk létre a logikai egységeket. De ezt majd csak a telepítés során tegyük meg, mert a DOS-os fdiskkel létrehozott partícióval a Linux alatt semmire sem fogunk menni. Legalábbis nem a telepítés szempontjából értékelhetően. Ha a fips nem vált be, vagy olyan partíciót kellene átméretezni, melyet az nem kezel, megpróbálkozhatunk a Partition Magic nevű kereskedelmi termékkel, mely sok feladattal megbirkózik, de egy biztonsági mentést nagyon ajánlunk előtte (Igaz, a fips használata előtt is!).

Oszd meg...

Ha a hely már megvan, a következő feladat eldönteni, hogyan oszthatjuk azt fel. A Linux a Unix-rendszerekkel azonos file- és könyvtárstruktúrát használ, azaz a könyvtárakat a „/" (jobbra dőlő vonal) választja el egymástól, és a rendszer gyökerét, azaz root-könyvtárát egyszerűen „/"-lel jelöljük. A rendszerben csak egyetlen gyökérkönyvtár van, minden ebből indul ki. Nem kell tehát a DOS alapú rendszerekből

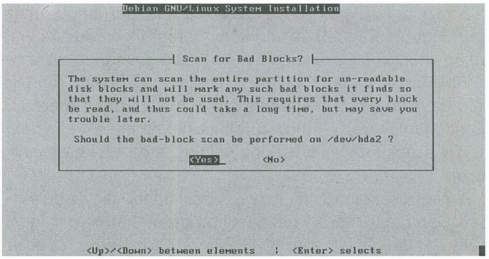


A swap partíció kiválasztása előtt

megszokott olyan típusú dilemmákra számítanunk, mint hogy vajon ma hogy hívják a C:-t. Természetesen ettől még szabadon használhatunk több lemezt, mobil egységeket és egy lemezen több partíciót.

Mindössze a csatlakoztatás elve változik meg: minden egyes egységet vagy partíciót a meglévő könyvtárstruktúra valamely pontjához kell illeszteni. Ez a pont néhány megkötéssel ugyan, de szinte bárhol lehet. Ez az elv teszi lehetővé számunkra, hogy felosszuk a lemezterületet, és külön partícióra kerüljenek a konfigurációs file-ok, külön a programok, külön a változó adatok és a felhasználóknak kiadott terület, de az ideiglenes tárolóhelyként használt könyvtár is kerülhet külön partícióra.

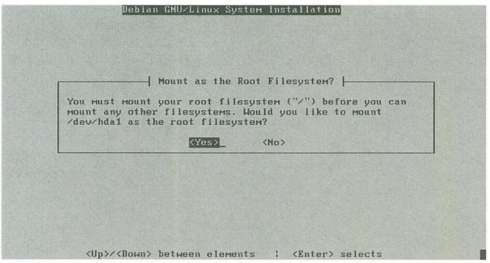
Ez a valóságban a következőképpen nézhet ki: az alapvető dolgok, melyek a rendszer indításához kellenek a „/"-be (erre ezentúl, mint „root" hivatkozunk), azaz a rootot tartalmazó partícióra kell, hogy kerüljenek. Meglepő lehet, de a kernel nem tartozik ezek közé. Ennek oka, hogy azt még a LILO (vagy más Linux indító, de ez a legelterjedtebb) tölti be, és miután elindult nem számít, hogy hol van. Kerülhet egy teljesen külön partícióra vagy RAM-diskbe töltődő file-ba. Mindkettőnek megvan a maga haszna.



Aki biztosra akar menni, ellenőrizheti, nincs-e hibás blokk a lemezen

A legfontosabb, hogy a LILO a BIOS-on keresztül elérhesse, azaz IDE diszkeken az 1023 cilinder alatti részre kerüljön. Debian alatt a kernel a /boot könyvtárban van, tehát ha a root nincs 1023 cilinder alatt, a /boot kerüljön külön partícióra, melyet a BIOS elér. A /boot számára annyi hely kell, hogy a kernelfile-ok elférjenek rajta. Egy-egy kernelfile 300–800 KB, azaz nem nagy. Ebből lehet egyszerre több is a rendszerben, így mi 6–8 MB helyet javaslunk a /boot számára. Ha a root partíció 1023 cilinder alatt van, vagy a BIOS biztosan eléri, a /bootot felesleges különrakni. A root partíció számára – ha mindent, amit csak lehet külön partícióra pakolunk – általában 32 MB bőven elegendő. Ennyiben elfér a rendszer alapja.

Mivel ide (pontosabban a /lib/modules könyvtárba, melynek a rootban kell lennie) kerülnek a kernel moduljai, ha sok különböző kernelt akarunk egyszerre telepíteni, érdemes a root partíciót 64 MB méretűre választani. Ennél több helyre nem lesz szükség.



A root file-rendszer kiválasztása

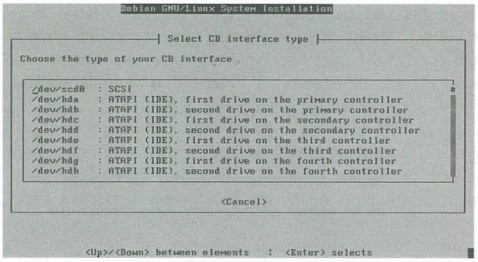
A programok és a libek, melyek a rendszer indulásához nem létfontosságúak, a /usr könyvtárba, illetve azon belül alkönyvtárakba kerülnek. Azaz ha újabb programokat telepítünk, azok leginkább innen vesznek el helyet. Ennek tükrében érdemes meghatározni a /usr partíció méretét. Gondoljunk arra is, hogy a felhasználóknak is kell majd helyet biztosítani, és a legtöbb helyigény ott szokott jelentkezni.

A nem csomagból, hanem a forráskódból vagy másképpen beszerzett, kézzel telepített programokat a /usr/local könyvtár megfelelő alkönyvtáraiba szokás rakni. Ez is az előbbi /usr-ben tartózkodik, így nem kell neki külön figyelmet szentelni, de jobb emlékezetben tartani létezését. A következő lépés, helyet biztosítani a /var számára. Ide kerülnek a leveleink, és szinte minden egyéb, a programok futása során változó adat – köztük a rendszer naplófile-jai. Ez a /var/log alatt kapott elhelyezést. Azt, hogy ez utóbbi a végtelenbe (és tovább – a szerk.) növekedjen, a rendszeresen lefutó karbantartó programok ugyan meggátolják, de így is szép méretűre duzzadhat. És akkor a leveleket még nem is említettük, amiket mindig mindenki a bejövő postaládájában, azaz a /var/spool/mail könyvtárban található felhasználó-specifikus file-ban felejt. Ha emellett még távoli hírcsoportokat is le akarunk tölteni, mindenképpen jó sok helyet kell hagynunk a /var számára.

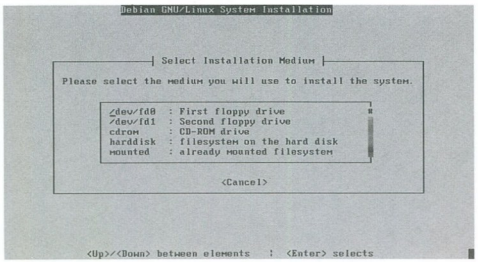
A felhasználóink által legtöbbet bitorolt terület a /home alatt lesz, ahol mindenki saját könyvtárat kap. Ez alapesetben a /home/azonosító, itt az „azonosító" a felhasználó loginneve, amivel belép a rendszerbe. Ha az adduser program (ezzel hozhatunk majd létre új felhasználót) beállításait átírjuk, a /home-ot picit átszervezhetjük. Két lehetőségünk is akad: a felhasználókat elsődleges csoportjuk alapján további alkönyvtárakba tehetjük, illetve az azonosító kezdőbetűje alapján is válogathatunk.

Mindkét esetben azt érjük el, hogy mivel /home alatt több külön könyvtárba vannak osztva, akár több külön partícióra is kerülhetnek az egyes csoportok tagjai. Ez a nagy felhasználószámmal rendelkező rendszereknél kifejezetten előnyös. Megmaradt még a /tmp, mely az ideiglenes file-ok tárolóhelye. Jegyezzük meg róla, hogy a rendszerünk minden induláskor letöröl innen mindent, valamint működése közben rendszeresen eltávolítja az egy hétnél régebbi dolgokat. Ez is kerülhet külön partícióra, így biztosan mindig állandó üres helymennyiség áll majd rendelkezésre. 64–128 MB-nál többet szerintünk semmiképpen sem érdemes erre a partícióra pazarolni. Egy alap Debian Linux rendszeren minden másnak a rootpartíción kell lennie. Ez alól kivétel a /opt, amit néhány üzleti termék „szeret", mint például az Applixware. Ez egy olyan könyvtár, ahol minden egyes alkalmazásnak külön könyvtára van, és ellentétben a /usr rendszerével – ahol egy-egy program részei szétszórva, külön a libek, külön binárisok kerülnek elhelyezésre –, itt minden együtt található. Ezt olyan jellegű programokhoz célszerű használni, ahol a felosztás nem kívánatos. Ha szándékunkban áll ilyeneket telepíteni, érdemes ennek is sok helyet biztosítani. Természetesen egy huszárvágással a teljes /opt-ot a /usr alá utasíthatjuk, ha létrehozunk például egy /usr/local/opt könyvtárat, és készítünk egy szimbolikus linket rá: a /opt-ba („mkdir /usr/local/opt; ln -s /usr/local/opt /opt").

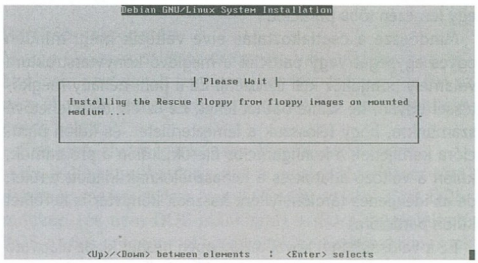
Ha valakinek a fenti még egyelőre túl bonyolult, nem kell aggódni, mehet a teljes rendszer egyetlen partícióra, a rootra. Csak arra kell figyelni, hogy a BIOS be tudja majd tölteni a kernelt. Erre a legegyszerűbb megoldás, ha a me-



CD-s telepítéskor a megfelelő CD-meghajtó kiválasztása



A telepítő médium kiválasztása



A legalapvetőbb file-ok telepítése

revlemezt LBA-módban használjuk, így minden része biztosan 1023 cilinder alatt lesz. Az előbbiekben leírt darabolást feltétlenül ajánljuk komoly rendszereken.

Ez nagyobb biztonságot jelent, mind a védelem, mind a működés szempontjából. Például teljesen felesleges, hogy a /usr írható legyen, ha már nem akarunk újabb programokat telepíteni, míg a /var-on felesleges megengedni a programfuttatást (ez alól kivétel a telepítés időszaka, mivel a Debian-csomagok telepítőszkriptjei innen futnak).

Ugyancsak felesleges a devicefile-ok engedélyezése a root partíción kívül bárhol. Ezek a különböző hardverelemek elérésére szolgáló speciális file-ok, és mindnek a /dev könyvtárban van a helye.

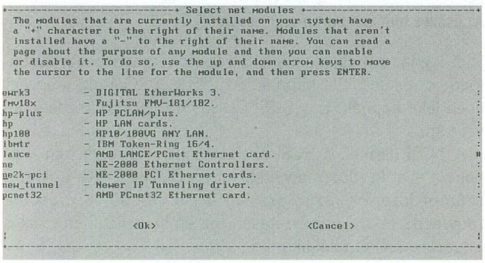
Mit – merre?

A fentiekben már tettünk említést arról, hogy mi hol fog elhelyezkedni a telepített rendszeren. Fontosnak érezzük azonban, hogy erről bővebben is szóljuk. A rendszer beállításai és az indítást végző szkriptek a /etc alatt helyezkednek el. A működő rendszerben találkozni fogunk a /proc könyvtárral, amely nem egy valódi, hanem egy speciális file-okat tartalmazó könyvtár. Ezek a file-ok adnak információt a futó rendszerről, illetve itt vezérelhetünk bizonyos kernel-funkciókat is. Például a futó programok listáját is a /proc-ból olvassa ki a ps program.

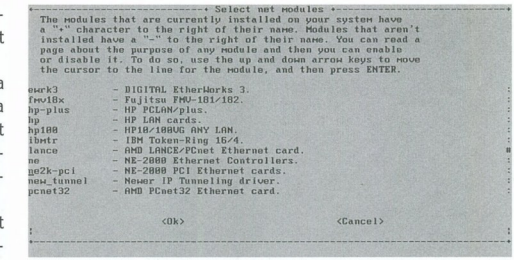
A rendszer indításához szükséges programokat a /sbin és a /bin könyvtárakban találjuk, az ezekhez szükséges libeket pedig a /lib alatt.

A /bin és /sbin közötti különbség csak annyi, hogy a /bin-ben a felhasználó számára is értékes programok találhatók, míg a /sbin-be kerülnek azok a programok, amikre csak a rendszernek és a rendszergazdának van szüksége. A /dev alatt vannak a hardvereszközök kezelésére szolgáló file-ok, a /boot alatt pedig a kernelek.

Találkozhatunk a fentieken kívül még néhány könyvtárral és file-lal a rootban, melyek nem létfontosságúak ugyan, de valamilyen oknál fogva itt kaptak helyet. Ilyen a /mnt, ami



A modulok beállítása: főmenü



A hálózati modulok (driver programok)

alá a külső file-rendszereket szokás csatolni, vagy az amd és az amnt.

Ezek különböző automatikus file-rendszer-csatoló programok számára vannak fenntartva. Találunk még egy szimbolikus linket a kernelre /vmlinuz néven. A LILO is itt keresi a kernelt, és ez mindig az aktuális kernelre mutat, ami ténylegesen a /boot-ban van. Az előző verziójú telepített kernelre /vmlinuz.old néven lelünk szimbolikus linket. Minden partíciónkon elhelyezkedik egy lost+found alkönyvtár. Ezt sose bántsuk, ha a file-rendszer ellenőrzésnél sehova sem köthető file-t talál a rendszer, azt ide teszi.

Szokás készíteni a CD-ROM-nak is egy könyvtárat /cdrom néven. Ha benézünk a /usr könyvtárba, ott is találunk bin, sbin és lib könyvtárakat.

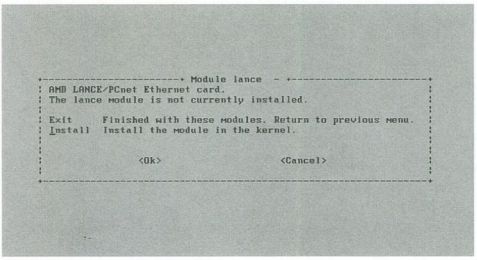
Ezeknek hasonló a célja, mint a rootban levőkének azzal a különbséggel, hogy ami itt van, az nem szükséges a rendszer elindulásához. Ide kerül továbbá a grafikus rendszer (X11R6), a csomagok és programok dokumentációját tartalmazó könyvtár (doc, info és man), a programok fordításához szükséges C fejléc (header) file-ok (include), az osztott információk (share), valamint a forráskódokat, elsősorban a kernel forráskódját tartalmazó könyvtár (src), illetve a már említett local.

A /var alatt számos változó anyag van, ennek tartalmát itt most nem részletezzük. Legfontosabb alkönyvtárai a spool, a lib, a lock, a run és a tmp. A spool alatt vannak a levelek, hírcsoportok és sok más mozgó anyag. A run alatt minden futó szolgáltatás futásazonosítóját (pid) találjuk egy-egy hozzájuk tartozó file-ban.

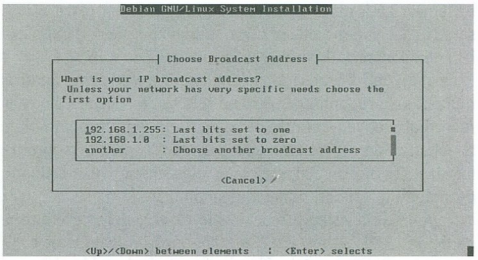
A lock az eszközök egyedi hozzáférésének biztosítására tárolja az állapotokat, míg a tmp a /tmp-hez hasonlóan egy ideiglenes tárolóhely.

Hogyan kezdjünk hozzá?

A telepítő-CD reményeink szerint bootolható, de nem ígérhetjük, hogy mindenkinek sikerülni fog erről kezdeni a telepítést. Alternatív megoldás, ha egy üres floppyra felírjuk a



Egy konkrét modul betöltése



A broadcast cím beállítása

telepítés indításához szükséges file-okat, melyeket a CD-n a boot könyvtárban találunk resc1440.bin néven. Ezt DOS alól a CD-n megtalálható rawrite programmal, Linux alól a dd paranccsal tehetjük meg. Ezen programok használata igen egyszerű. DOS esetén indítsuk el a dosutils könyvtárban található rawrite.exe programot, majd először adjuk meg neki a kiválasztott file-t (resc1440.bin), ezután a lemezegységet, ahova fel akarjuk írni (a:). Vigyázat! A lemez korábbi tartalma megsemmisül. Hasonlóan óvatosan bánjunk a linuxos dd programmal, melyet az alábbi módon érdemes indítani: „dd if=resc1440.bin of=/dev/fd0", ha abban a könyvtárban állunk, ahol a resc1440.bin file is található. Értelemszerűen a példákban szereplő resc1440.bin file helyett használjuk azt, amelyikre szükségünk van. A telepítés pontosan ugyanúgy indul, függetlenül attól, hogy azt a CD-ről vagy floppyról kezdtük. Első lépésként bejelentkezik a syslinux, mely számos információval lát el bennünket. A telepítéshez itt egyszerűen nyomjuk meg az ENTER-t, vagy ha valami speciális ok miatt kernel paramétereket kell megadnunk, azt tegyük meg a képernyőn leírtak alapján.

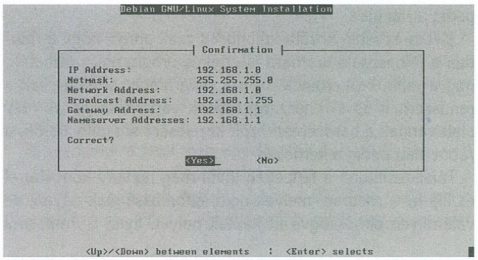
A telepítés menete

Alapbeállítások

Az első lépés néhány alapvető beállítás elvégzése, és a rendszerinformációk elolvasása. Ezek közé tartozik, hogy eldöntsük, színes-e a monitor, illetve milyen billentyűzetet használunk. Ha ezzel megvagyunk, elértük a telepítés főmenüjét. Itt értelemszerűen szabadon választhatunk a feladatok közül, de a rendszer mindig felajánlja, hogy mi legyen a következő lépés, illetve igen gyakran alternatívát is felkínál. Ez utóbbi például akkor érdekes, ha már van linuxos partíció, így nem érzi szükségesnek particionálni, de mi tudjuk, hogy kell. Ilyenkor az alternatív lépést, azaz a particionálást választva megtehetjük azt. Ezután folytathatjuk tovább a lépéseket, ahogy eddig. Nem javasoljuk, hogy bárki is megszakítsa a telepítés menetét, kivéve, ha erre nyomós oka van. A legtöbb szükséges vezérlő – itt elsősorban a SCSI vezérlőkre gondolunk – drivere benne van a telepítő kernelben, így ezzel itt nem kell külön törődni. Ha mégsem, a CD dists/slink/main/disks-i386/current/ könyvtárában találunk néhány alternatív indítólemezt (mind resc1440-nel kezdődik). Ha az sem segít (ez nem valószínű), kérjünk meg valakit, hogy készítsen egy egyedi kernelt, és a boot-floppies csomag segítségével készítsen új telepítőt.

Particionálás

Tehát az első lépés a merevlemez particionálása, hogy helyet csináljunk a root-partíciónak és a swapnek. Az IDE lemezek /dev/hd, a SCSI merevlemezek /dev/sd kezdetű nevet kapnak. Ezután következik egy betű, mely gyakorlatilag „sorszámozza" a lemezeket. IDE esetén az első IDE vezérlő elsődleges egysége az „a" jelet, a másodlagos egysége a „b"

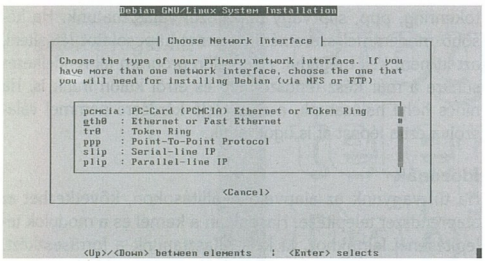


Az összes hálózati paraméter egyben

jelet kapja, a másodlagos IDE vezérlő lemezei pedig hasonlóan „c" és „d" jelet kapnak. Az IDE CD-ROM-ok is ugyanilyen jelet kapnak. Egy SCSI CD-ROM /dev/scd kezdetű nevet kap.

A SCSI diszkek a SCSI azonosító (ID) sorrendjében kapnak „a"-val kezdődő neveket, míg a SCSI CD-ROM-ok sorszámot kapnak, azaz az első a /dev/scd0, a második /dev/scd1 lesz. Természetesen, az említett listában csak a merevlemezeket találjuk meg

Válasszuk ki a kívánt egységet. Később még lesz módunk



Az elsődleges hálózati csatoló kiválasztása

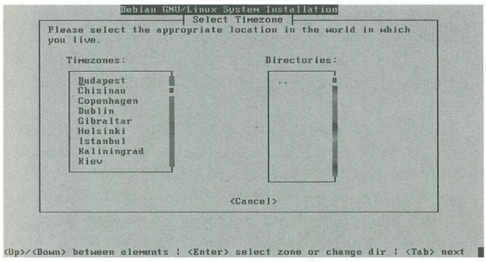
más egységek particionálására is. Ha ez megvan, elindul a cfdisk program, mely egy egyszerű, menüs particionáló. Használata egyszerű: a fel- és le- nyilakkal léphetünk a bejegyzések között, a „d" betűvel törölhetünk partíciót, az „n"-nel hozhatunk létre újat, a „t" változtatja meg a típusát (ne felejtsünk el létrehozni Linux-swap, azaz 82-es típusú partíciót).

Ne feledjük, maximum négy primary, vagy három primary és több logikai partíciót hozhatunk így létre. Ha bármit elrontunk, a kilépés (Quit) eldobja a változtatásokat és kezdhetjük elölről a folyamatot.

A root-partíciót végül állítsuk bootolhatóvá (bootable), hogy a rendszer elindulhasson onnan. Ha kész a particionálás, írjuk fel a lemezre a partíciós táblát (Write, majd a kérdésre „yes").

A következő lépés a swap-partíció kiválasztása. Ismét egy listából kell kiválasztani az alkalmazni kívánt swapterületet. A lista csak akkor többelemű, ha több linuxos swappartíció is létezik.

Ha szükséges, ellenőriztethetjük a rendszerrel, hogy van-e hibás blokk a merevlemez ezen területén. Új diszkek esetén erre nincs szükség. Ezután a linuxos natív partíciók formázása következik. Hasonlóan az előbbiekhez itt is egy listából kell kiválasztani a formázni kívánt partíciót. Először mindig a root-partíciót formázzuk le, és az ezt követő kérdésre, mely arról érdeklődik, hogy ez tárolja-e a root file-rendszert, válaszoljunk igennel!

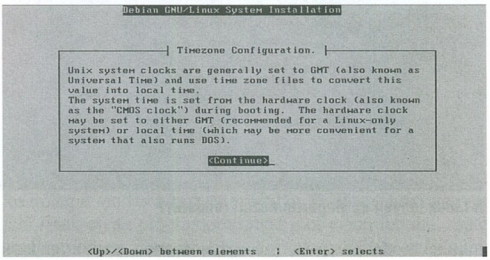


Az időzóna kiválasztása

Ha több partíciót is létrehoztunk a rendszerünkhöz, a főmenübe visszaérve a következő helyett válasszuk a Linux partíció formázása alternatív lehetőséget, és sorban formázzuk le az összes új, neki szánt partíciót. Minden egyes alkalommal meg fogja kérdezni a telepítő, hogy a file-rendszeren belül hova kívánjuk csatolni a frissen formázott partíciót. Ha már van előzőleg formázott partíciónk, azt is csatolhatjuk egy másik alternatív lépést választva.

Kernel

Ezután a kernel és az operációs rendszer alapjainak telepítése következik, miután a telepítő kérdésére válaszolva jóváhagytuk a file-rendszer felosztását. Ki kell választanunk azt az eszközt, melyről telepíteni kívánunk. Ez lehet valamelyik floppymeghajtó (ez esetben további nyolc floppyra lesz szükségünk, melyek ugyanott vannak, ahol az alternatív telepítőlemezek, base14 kezdetű névvel, és ugyanúgy lehet floppyra írni őket), CD-ROM, merevlemez vagy egy már csatolt file-rendszer is. Itt még nem kérhetjük a hálózati telepítést, ahhoz a most sorra kerülő alaprendszernek már mű-



GMT vagy nem GMT...

ködnie kell. Mi most a CD-ROM-ról történő telepítést követjük. Kiválasztása után egy újabb listával találjuk szemben magunkat, ahol a megfelelő CD-ROM-egységet kell kiválasztani. A bevezetőben már volt szó róla, hogy melyik milyen néven érhető el, így most ez alapján kell kiválasztanunk a megfelelőt. Ne felejtsük el betenni a Debian első CD-jét a meghajtóba!

Ha a telepítő sikeresen csatolta a Debiant tartalmazó lemezt, megkérdezi, hol találhatók rajta a szükséges file-ok. Ez a hivatalos és az általunk készített CD-n is a /debian könyvtár, ami egyébként az alapértelmezés is.

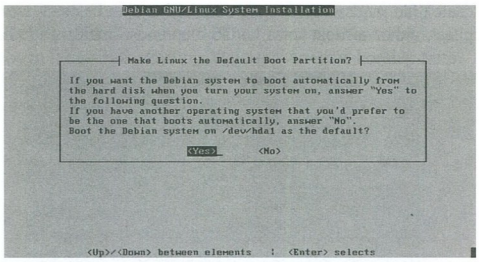
A következő választási pontban kérhetünk egy listát az elérhető alaprendszerekből – mert több is lehet, de az általunk készített CD-n csak egy van –, vagy választhatjuk, hogy kézzel adjuk meg az alaprendszert tartalmazó file nevét. Mi a listából választást javasoljuk. A telepítő ezután felteszi a kernelt és a modulokat (azaz a drivereket). A következő lépés a hardver konfigurálása, pontosabban a modu-

lok beállítása. Amelyik modult itt betöltetjük a rendszerrel, az minden későbbi indításkor (abban az esetben, ha a /etc/modules file-t nem módosítjuk) azonos paraméterezéssel kerül betöltésre.

Modulok

Szükség esetén (például noteszgép telepítésekor) alternatívaként itt állíthatjuk be a PCMCIA szolgáltatásokat is. Csak azokat a modulokat töltsük be, melyekre feltétlenül szükség van a rendszer mindennapos működésekor. Ilyen lehet például a hálózati kártya drivere, melyet a „net" csoportban találunk, vagy a „block" csoportból a különböző RAID-funkciókhoz szükséges driverek.

Gyakorlott Linux-felhasználók itt hozhatják létre a szükséges RAID-tömböket, hogy a telepítés már azokra történjen. Itt most csak a tájékoztatás szintjén említjük meg, hogy a Linux képes RAID0, 1, 4 és 5 szoftveres kezelésére. Mivel



A Linux legyen az alapértelmezett rendszer?

minden modul paraméterezése más és más, csak a rendszer által adott információkra hagyatkozhatunk ezek megadásakor. Annyiban biztosak lehetünk, hogy a PCI-os eszközök moduljainak a legritkább esetben kell paramétert megadni, hiszen a PCI tényelegesen PnP eszköz. A legtöbb esetben tehát érdemes paraméterek nélkül tenni egy próbát. Ha ez nem sikerül, újra meg lehet próbálni paraméterek megadásával is. Ha minden szükségesnek ítélt modult betöltöttünk, lépjünk ki a hardver konfigurációs menüből. Ha netán elfelejtettünk valamit semmi gond, később a működő rendszerben hozzáadhatjuk.

Hálózat

A hálózati alapkonfiguráció a soron következő lépés. Itt először olyan általános, és feltétlenül szükséges adatokat kell megadnunk, mint gépünk egyedi neve, majd attól függően, hogy rendszerünk kapcsolódik-e helyi hálózathoz, ennek adatait, azaz a tartománynevet (domain name), IP-számot, netmaskot, broadcast címet, alapértelmezett átjárót (gateway), név kiszolgálót (DNS), valamint a kapcsolatról gondoskodó hálózati csatoló fajtáját. Itt PCMCIA, ethernet, tokenring, ppp, slip vagy plip közül választhatunk. Ha később modemmel szeretnénk Internet-kapcsolatot létesíteni, azt itt nem kell megadni, erre külön program áll rendelkezésünkre a már kész rendszerben, és erről külön írunk is. Ha nincs helyi hálózatunk, a megfelelő kérdésre nemmel válaszolva ezt a lépést át is ugorhatjuk.

Időzónák

Ha túl vagyunk az alapvető beállításokon, következhet az alaprendszer telepítése. Hasonlóan a kernel és a modulok telepítésénél leírtakhoz, ki kell választanunk a forráseszközt. Ha floppyról telepítünk, itt lesz szükség a további hét lemezre. Ha CD-ről installálunk, az alaprendszer telepítése egyvégtében megtörténik. Miután ezzel is elkészültünk, hátra van még néhány beállítás. Ezek közé tartozik az időzóna (Magyarországhoz Europe-ot, majd Budapestet javasoljuk) kiválasztása és annak eldöntése, hogy a rendszeróra GMT (Greenwich Mean Time – greenwichi idő) szerint járjon-e. Ha csak Linuxot használunk a gépen, ez utóbbi javasolt, mert így a nyári és téli időszámítás-váltás automatikus, miközben a hardverórához nem kell nyúlni, azaz biztosan konzisztens marad a rendszeridő.

LILO

A végső lépés, hogy indíthatóvá tegyük a telepített rendszert. A Debian Linux telepítője a LILO-t, azaz a Linuxot indító programot (LInux LOader) a root file-rendszer elejére teszi. Ahhoz, hogy ott elinduljon, ennek kell lennie az első aktív partíciónak, és a diszk MBR-jéből olyan programnak kell indulnia a gép bekapcsolása után, amely az első aktív partíciónak átadja a vezérlést. A merevlemezek gyári bootprogramja ilyen, de ilyen a DOS- és Windows-rendszereké is, így ez nem okozhat problémát.

Amennyiben szükséges, itt választhatjuk ki azt is, hogy alapértelmezés szerint frissen telepített Linux rendszerünket indítsa-e a LILO. Abban az esetben, ha már egyszer egy másik LILO felülírta az MBR-t, gondban leszünk, de a megoldás nem nehéz. Vagy fogunk egy DOS-rendszerlemezt, és a DOS-os fdisk parancsot „fdisk /mbr" opcióval lefuttatjuk, vagy floppyról indítva a kész rendszerünket beállítjuk a /etc/lilo.conf-ban, hogy a LILO a merevlemez MBR-jébe kerüljön.

Legvégül készíthetünk egy bootlemezt is, mely a beállításaink alapján elindítja a telepített rendszert. Ezt mindenképpen érdemes elkészíteni, mert bármikor hasznunkra válhat.

Az utolsó lépés a rendszer újraindítása. Mielőtt ezt a menüpontot választanánk, ne felejtsük el kivenni a bootlemezt, vagy ha CD-ről indítottuk a telepítést, akkor a CD lemezt. Ha mindent jól csináltunk, rövidesen megjelenik egy LILO felirat, majd néhány másodperc elteltével automatikusan elindul új rendszerünk, ahol folytathatjuk a konfigurálást és telepítést. ■

Slapic
slapic@vogel.hu

Programok telepítése Debian alá

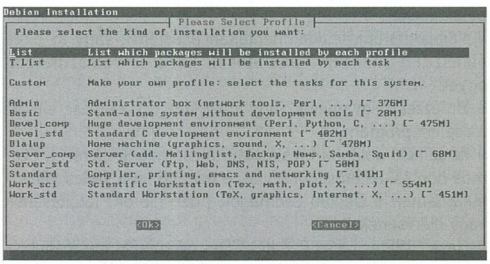
A Debian alapjának telepítésével még korántsem készítettünk egy működő rendszert, így azt nem is tekinthetjük a telepítés végének. Most itt folytatjuk az alaprendszer néhány beállításával, a csomagok kiválogatásával és telepítésével.

Ha a telepítés során minden jól ment, az újraindítás után megjelenik a LILO, majd látjuk elindulni a kernelt.

Ha a rendszerünk elindul, hamarosan kéri, hogy állítsuk be a root, azaz a rendszer mindenható urának jelszavát.

Válasszunk bonyolult, nehezen megfejthető jelszót, hiszen ez véd meg bennünket attól, hogy illetéktelenül hozzáférhessenek a rendszerhez. A jelszót kétszer kell megadni a biztonság kedvéért. A beírt jelszót nem lehet visszakeresni, így azt jól jegyezzük meg, de ne írjuk le olyan helyre, ahol más is hozzáférhet. A legjobb nem leírni. Ha netán mégis elfelejtenénk, és nem tudunk bejutni a rendszerbe, nem kell elkeseredni, van megoldás, de ezt máshol ismertetjük.

A következő lépés egy felhasználó létrehozása. Ezt a lépést ugyan ki lehetne hagyni, de mi (és a Debian készítői) nem javasoljuk. Nagyon fontos alapelv a Unix típusú rendszerek – amilyen a Linux is – üzemeltetésénél, hogy a rootfelhasználót kizárólag a rendszer karbantartásához, beállításához használjuk. Csak olyan programot szabad root jogon futtatni, mely az alaprendszer része és meg vagyunk győződve arról, hogy megbízható. Ha ezt betartjuk, a rendszerünk biztonságán sokat javítottunk, és elejét vettük a potenciális vírusok vagy a trójai programok terjedésének. Minden olyan feladatot, melyhez nem kell a rootfelhasználó egyszerű userként végezzünk el. Erre jó, ha már most létrehozunk egy ilyet. Később további felhasználókat is hasonló módon adhatunk hozzá a rendszerhez az adduser paranccsal. Meg kell adni a felhasználó nevét, jelszavát (itt is kétszer), valamint néhány személyes adatot: valódi név, telefonszám, munkahely. Ez utóbbiakat persze nem kötelező kitölteni, és a chfn paranccsal a későbbiek során bármikor módosíthatjuk. Jelszót később passwd paranccsal változtathatunk.



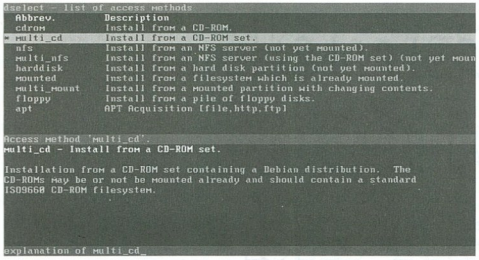
Az általános feladatkiválasztás

```
in cases that you need special privileges (such as when you need to
mount a file system). Now you may create a normal user account.

Note that you may create it later (as well as any additional account)
by typing 'adduser <username>' as root, where <username> is an
(eight-character or less) user name, like 'imurdock' or 'rms'.

Shall I create a normal user account now? [Y/n]
Enter a username for your account: slapic

Adding user slapic...
Adding new group slapic (1000).
Adding new user slapic (1000) with group slapic.
Creating home directory /home/slapic.
Copying files from /etc/skel
Changing password for slapic
Enter the new password (minimum of 5, maximum of 8 characters)
Please use a combination of upper and lower case letters and numbers.
New password:
Re-enter new password:
Password changed.
Changing the user information for slapic
Enter the new value, or press return for the default
        Full Name []: _
```

Egy általános felhasználó felvitele telepítéskor

A programok kiválasztása

Ha a fentieken túlvagyunk, lehetőségünk nyílik még a dselect nevű csomagkezelő előtt általános beállításokat elvégezni arra nézve, hogy mire használjuk a rendszert, milyen csoportokba tartozó programok kerüljenek fel.



A telepítés forrásának meghatározása

Ha kérjük ezt, egy menübe jutunk, ahol elvégezhetjük ezen beállításokat.

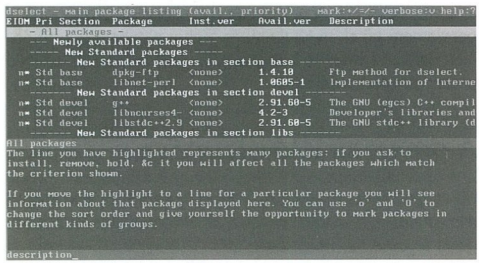
Válogathatunk feladatorientáltan (Custom), vagy az előre elkészített „konzerv" beállítások közül is választhatunk egyet.

Ha ezzel megvagyunk vagy kihagytuk ezt a lépést, elindul a dselect program.

Itt az első dolgunk kiválasztani a csomagok elérését. Számos döntési lehetőségünk van, a telepítő igen széles választékát ismeri a Debian csomagok (.deb file-ok) letöltésének vagy elérésének.

Ha a CHIPtár Linux 2.2 két CD-s csomagját használjuk, a multi_cd opcióval adhatjuk a rendszer tudtára, hogy a telepítő több mint egy CD-n foglal helyet. Ha nem, itt választhatjuk a hálózati telepítést NFS-ről vagy egy helyi diszkről, vagy bármilyen már csatolt eszközről, illetve hálózati driveról. Ha FTP- vagy WWW- (HTTP)-szerverről szeretnénk telepíteni, az apt-t válasszuk. Ez a program ugyan még fejlesztés alatt áll, de az említett két (és a helyi file-elérésen alapuló) telepítési metódusa már stabil és használható. A későbbiekben ez lesz az egyetlen csomagkezelő, mai fejlesztői változata (mely ezen a CD-n nem kapott helyet), amely már ismeri a több CD-s telepítést is.

Tehát a több CD-s telepítéshez először a második CD-t tegyük be, válasszuk a multi_cd opciót, majd adjuk meg a CD-eszköz nevét. Ez most az alapértelmezett /dev/cdrom,



Csomagok kiválasztása dselect-ben

így egyszerűen Entert kell ütnünk. Lehetnek olyan részek, melyeket a telepítő keresni fog, de nincs a CD-ken. Ilyen például a „local" csoport. Ez nem hiba, egyszerűen jelezzük egy „none" beírásával, hogy ilyen nincs, ne is keresse. Az általunk készített CD-ken található local rész, így ennél nem találkozhatunk ezzel. A telepítés hibátlanul lefut nélkülük. Ha sikerült beállítanunk az elérést, visszakerülünk a főmenübe, ahol jöhet a második, azaz „Update" menü, mely a megadott helyről frissíti az elérhető csomagok listáját.

Válogatott portéka

Miután frissítettük az adatbázisunkat, jöhet a programok kiválogatása. Ide akkor is lépjünk be, ha az előzetes kiválasztás alapján kérjük a telepítést, és nem akarunk itt semmi mást megváltoztatni. Erre a függőségkezelés miatt van szükség, de erről picit később. Mivel a dselect egy hatékony, de nem túl felhasználóbarát program, itt szólnunk kell néhány szót a használatáról. Mindenekelőtt megemlítjük, hogy a CD-n, ugyanott, ahol az indítólemez és a base rendszer file-jai is vannak, találunk egy igen jó leírást a programról (dselect-beginner.txt). Ez a program nagyon sokat örökölt a unixos elvekből, melyeket ha megismerünk és megértünk kifejezetten kellemessé teszik a használatát. Szinte bármikor kérhetünk segítséget a ? (kérdőjel) leütésével.

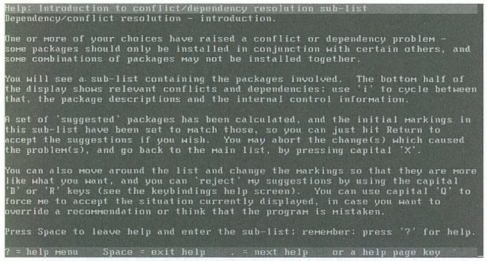
Ha beléptünk a csomagok kiválasztásához, először egy információs oldalt látunk, ahonnan (és ez igaz majdnem minden információs oldalra a dselectben) a szóköz leütésével léphetünk tovább. Minden esetben olvassuk el a tájékoztatót, fontos információkat közöl benne a program. Ha ezen túléptünk, a csoport- és csomaglistában találjuk magunkat. Itt pár egyszerű billentyűvel mozoghatunk és dolgozhatunk: a kurzormozgató fel- és le nyilakkal, illetve a PageUp és PageDown gombokkal lépkedhetünk a listában, mely alapértelmezés szerint tagolt. Először jönnek az új csomagok, melyek az előző használat óta jelentek meg az archívumban (ennek a CD-ről történő első telepítéskor nem sok jelentősége van), majd a többi csomag, szekciónként, ahogy azok a CD-n is könyvtárakba vannak osztva.

Csomagtan

Egy-egy csomaginformációt tartalmazó sor alapfelépítése a következő: négy karakternyi állapotjelző, minősítés, csoport, csomag neve, telepített csomag verziója, elérhető csomag verziója és végül egy rövid leírás. Itt az első négy karakternyi rövid állapotjelzésből láthatjuk, hogy telepítve van-e, ki van-e jelölve telepítésre, esetleg hibásan van telepítve, vagy telepítve van ugyan, de még nincs konfigurálva. A kért állapoton az „I", „-" és „_" gombokkal tudunk változtatni. Az „I"

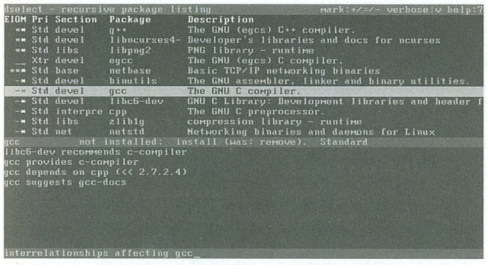
vagy az „Ins" gombok kijelölik telepítésre, a „-" törlésre állítja, a „_" megsemmisítésre. Ez utóbbi abban különbözik a sima törléstől, hogy az esetleges beállításokat is eltávolítja, míg a sima törlés nem. Ha problémánk akadt egy csomaggal, az „=" egy olyan állapotba teszi, hogy a csomagkezelő ne változtasson a meglévő állapotán („hold"). Ezt csak végszükség esetén alkalmazzuk, azaz a CD-ről telepítéskor NE. Keressünk is a listában, ha leütjük a „/" jelet, majd begépeljük a keresendő szövegrészt, végül nyomunk egy Entert.

Előfordul, hogy abban a pillanatban amikor módosítjuk a kért csomag valamelyik állapotát, egy információs képernyő

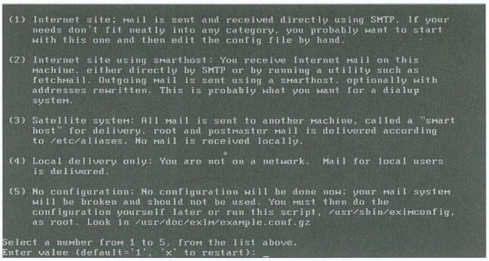


Függőségi probléma tájékoztató lap

jön fel, majd ha abból a szóközzel kiléptünk, egy újabb kiválasztás kép, mely eltér attól, ahol dolgoztunk. Ha figyelmesen elolvastuk az információt kiderül, hogy a váltás oka függőségi probléma. Ez pontosabban azt jelenti, hogy a kiválasztott program működéséhez más csomag telepítése is kell, vagy a törlésre jelölt program kell mások működéséhez. Az új ablak (melyből ha végeztünk, pontosan oda térünk majd vissza, ahonnan indultunk) ezt a problémát segít rendezni. A beállításaival azonnal ajánl egy lehetséges megoldást (ha van), így sokszor elég egy Entert ütni, hogy visszatérjünk, bár ez is eredményezhet újabb függőségi kérdést. Jó esetben több Enter visszavisz, rossz esetben végig kell gondolni, mit és hogyan szeretnénk, és néhány változtatást kell végrehajtani ahhoz, hogy megoldódjon a probléma. Ha elkészültünk minden



Függőségi probléma javasolt megoldása



A levelezőszerver alapbeállítása

beállítással, szintén az Enter megnyomásával léphetünk vissza a főmenübe. Itt is előjöhet a már említett függőségi ütközés megoldására szolgáló kép. A megoldás itt is ugyanúgy történik.

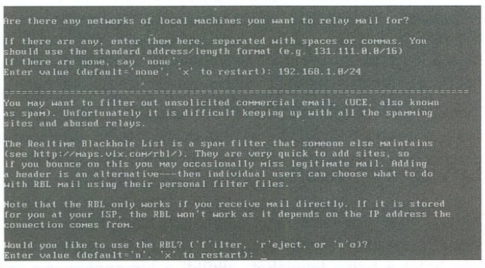
Visszatérve a főmenübe az „Install" opcióval megkezdhetjük a telepítést. Ekkor a dselect meghívja a dpkg-t, mely a beállítások alapján végignézi az elérhető csomagokat, és egyenként eldönti, hogy felrakja-e. Ez egy hosszadalmasabb folyamat, nyugodtan hátra lehet dőlni és várni. Ha túl van a telepítésen, elkezdi beállítani a feltett csomagokat („Setting up..."). Ekkor már igen gyakran fog feltenni kérdéseket, melyek megválaszolásához sokszor alaposabb ismeretre van szükség. Ennek hiányában mindig az alapértelmezett választást javasoljuk. Egy jó példa az ilyen kérdésekre a levelezőszerver beállítása. A 2.1-es Debianban az Exim az alapértelmezett levelezőkiszolgáló, de helyette természetesen választhatjuk az smailt, sendmailt vagy a qmailt (lásd a cikket) is, hogy csak az ismertebbeket említsük.

Levélügynökök

Ahhoz, hogy a levelezésünk elinduljon, egy jól konfigurált smtp-szerverre van szükség, mely továbbítja rendszerünkön belül a belső leveleket, elküldi a kimenőket, és ha szükséges, fogadja a bejövőket. A beállítás könnyű és gyors, ha tudjuk a megfelelő válaszokat, ami jelen esetben egyértelműen következik abból, mire akarjuk használni a rendszert. Az Exim beállításánál az első kérdés pontosan arra vonatkozik, mit is akarunk a programtól. Ha internetes szerverünket telepítjük éppen, válasszuk az 1. pontot, mely teljes TCP/IP levelezést jelent. Otthoni, illetve offline gép esetén a 2. pont előnyösebb, mivel ott megadhatunk egy olyan levelezőszervert (smarthost), mely fogadja és továbbítja az összes kimenő levelünket. A 3. pont műholdas kapcsolathoz, a 4. a hálózat teljes hiánya esetén hasznos, míg az 5. pontot választva elhalaszthatjuk a beállítást (Vigyázat! Ez esetben semmiféle levelezésünk nem fog működni!).

A következő lépés a rendszerünk látható nevének megadása. Ez a név fog szerepelni minden kimenő levelünk From: sorában a @ után. Ha ezen kívül más címre érkező leveleket is akarunk fogadni, például a gépünk neve debian.vogel.hu, de a mail.vogel.hu címre jövő leveleket is helyben akarjuk kézbesíteni ezt a következő lépésben megadhatjuk. Ha nincs ilyen cím, válaszoljunk „none"-nal. Be kell még állítani, hogy mely címek számára működjünk levéltovábbítóként (relay). Ezt ténylegesen korlátozzuk a szükséges mértékben. Egy átlag rendszer egyáltalán nem továbbít más idegen rendszernek levelet, csak a saját leveleivel foglalkozik, de ha a cégünk levelezőszerveréről van szó, melyen keresztül a kollegáink is leveleznek már engedélyezni kell a továbbítást. Ezt úgy a legegyszerűbb beállítani, hogy egy adott alhálóról jövő kapcsolatok számára lehetséges a továbbítás, másoknak pedig nem.

Esetleg megadhatjuk azt, hogy minden olyan rendszer számára, akihez be agyunk jegyezve, mint MX – azaz levéltovábbító-rendszer – továbbítunk levelet. Ez viszont azzal a kockázattal jár, hogy bárki irányíthatja hozzánk a saját címét a saját név-kiszolgálóján. A semminél viszont sokkal jobb. Erre azért van szükség, mert a kéretlen üzleti hirdetéseket, azaz spamet küldők az ilyen nyitva felejtett kapukat használják leveleik továbbítására. Ez eleinte csak bosszantó lesz számunkra (főleg a sértett rendszergazdák panaszáradata),



További beállítások a levelezőszerverhez

de idővel a címünk kitiltásával fog járni a legtöbb rendszerből, amit mégsem akarhatunk. Több nemzetközi adatbázis van, melyekre támaszkodhatunk, így megvédhetjük magunkat a nem kívánt levelektől. Az egyik – talán legismertebb – a VIXIE RBL (http://maps.vix.com/rbl/). Ennek használatát is bekapcsolhatjuk itt.

Emellett ha feltesszük a spamdb csomagot, annak leírásában megtaláljuk azt, hogy hogyan használjuk ki az általa rendszeresen frissített adatbázist.

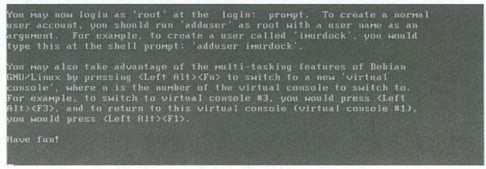
Ha válaszoltunk az Exim beállítószkript minden kérdésére, a végén összegzi a beállításokat, amit vagy elfogadunk, vagy újra végigmegyünk (az előzőleg beírt adatokat alapul véve) a beállításon.

Ha a beállítás megfelelt, és elfogadtuk, máris van egy működő levelezőszerverünk. Ugyanilyen egyszerűen bírhatjuk működésre a sendmail vagy smail programokat is, ha mégsem az Eximet választanánk.

Legkisebb közös többszörös

Ha több programot választottunk ki, sok hasonló procedúrán kell még végigmennünk, mielőtt a telepítés befejeződne. Ha gyorsan akarunk egy működő alaprendszert, csak annyit válasszunk ki, amennyi ehhez kell, majd a már felállt rendszerben a dselectet root-ként indítva ugyanúgy telepíthetünk, frissíthetünk vagy távolíthatunk el csomagokat, mint legelőször.

Egyet ne feledjünk: ha itt túl vagyunk a telepítésen és a kérdéseken, a program működik, ahhoz már csak akkor kell hozzányúlni, ha finomhangolást szeretnénk elvégezni, vagy



A telepítő utolsó üzenete: az új rendszer üzemel

speciális beállításokra van szükségünk, melyet a beállítószkript nem tudott megcsinálni. Igaz ez a grafikus felületre, azaz az X11-re is. A telepítő menetközben többször is megkérdezi (ha még nincs), hogy létrehozza-e a konfigurációs file-t a grafikus felület működéséhez.

Erre csak akkor válaszoljunk igennel, ha addigra már minden szükséges komponens telepítésre került. A legbiztosabb a telepítés után root-ként belépni, és elindítani az „xbaseconfigure" parancsot, mely ezt elvégzi. Ez a jól ismert XF86Setup programot indítja, melynek kezelését a CHIP Magazinban már részletesen leírtuk.

Ha a telepítés és a konfigurálás sikeresen befejeződött, megjelenik az „Installation OK." felirat, és az Enter megnyomásával visszakerülünk a főmenübe.

Itt találunk még pár opciót, mint „Configure" és „Remove". A „Configure" gyakorlatilag a „dpkg –configure –pending" parancs kiadása, mely a még valamilyen okból (például hibával megszakadt telepítés) nem konfigurált csomagokat próbálja meg beállítani, míg a „Remove" a törlésre kijelölt csomagok eltávolítását végzi. Ha már semmi dolgunk a dselectben, a „Quit" menüpontot választva kiléphetünk. Ha most telepítettük a rendszert egy üdvözlőszöveg után megjelenik a „login" prompt.

Belépheünk a rendszerbe, és elkezdhetünk dolgozni. ■

Kiadványunkban megpróbálunk mindenkinek segíteni, aki számítógépét házimozi vagy házi filmstúdió központjává szeretné formálni.
Mivel a számítógépes filmezés esetében még a legalapvetőbb technológiák is legfeljebb néhány évesek, különös gondot fordítottunk arra, hogy megismertessük Olvasóinkat ezekkel.
A gyakorlatiasabbak kedvéért pedig megvizsgáltuk a témához kapcsolódó hardvereket és szoftvereket, és leírtuk, mit találtunk. A legújabb fejlemény a DVD, aminek esetében már csak nagyon alapos vizsgálatokkal deríthető ki, hogy hol ér véget a számítógép-, és hol kezdődik a filmipar.
A tesztek remélhetőleg mindenkinek segítenek a céljaikra legmegfelelőbb befektetés kiválasztásában.

Nem lehetne teljes egy a multimédia világával foglalkozó kiadvány CD-melléklet nélkül, hiszen nem elég beszélni a képekről és a hangokról, illik meg is mutatni őket.

VOGEL

Kis Gergely
kisg@lme.linux.hu,
Bán Szabolcs
bansz@szif.hu

GNOME

A unix stílusú rendszerek ellen szóló egyik érv az volt, hogy nem állt rendelkezésre egy egységes grafikus felület, melynek segítségével a felhasználó bármelyik ugyanolyan módon használhatta volna.
A létező grafikus programok nagy része más-más felülettel rendelkezett, ez megnehezítette a megtanulásukat.

Többször is próbálkoztak az egységesítésükkel. Ezek közül kiemelkedő kísérlet a Common Desktop Environment (CDE), ami viszonylag sok unixváltozatban elterjedt.

A Linux-rendszerek megjelenése sem változtatott sokat ezen a képen, hiszen jó ideig nem készült hozzá X Window Systemre épülő egységes grafikus felhasználói felület (GUI). Csupán a CDE-t írták át Linuxra, ami viszont drága volt, és nem felelt meg a linuxos társadalom igényeinek. Ez a helyzet az elmúlt években jelentősen megváltozott. Néhány hamvába holt próbálkozás után megindult a K Desktop Environment (KDE), majd később a GNOME fejlesztése. Közös a két projektben, hogy nemcsak Linuxon, hanem más unixváltozatokon is futnak.

A GNOME egy mexikói egyetemista, Miguel de Icaza Amozurrutia ötlete alapján készült. Jelenleg is ő irányítja és hangolja össze a nemzetközi programozócsoport munkáját.

A GNOME (GNU Network Object Model Environment) tehát nem más, mint egy új, egységes grafikus felhasználói felület készítését célzó project. A KDE-hez hasonlóan az egységes kinézeten kívül programozói felületet is biztosít annak érdekében, hogy az egyes programok egymással jól együttműködve futhassanak. A két projekt munkatársai eldöntötték, hogy a lehető legnagyobb mértékben együttműködnek, így a KDE és a GNOME alá írt programok képesek lesznek kommunikálni egymással. A szabad szoftverek világára jellemző, hogy ugyanazt a feladatot több program is képes elvégezni, így mindig megvan a lehetőségünk arra, hogy az adott körülményeknek legjobban megfelelő megoldást válasszuk.

Ez a két projekt jó példa erre. A GNOME-nak van néhány olyan tulajdonsága, mely a KDE-ben nem, vagy másképpen található meg, viszont olyan nagy űrt töltenek be a szoftverpiacon, hogy képesek egymás akadályozása nélkül fejlődni.

A GNOME teljes mértékben szabad szoftverekre épül, míg a KDE-ről ez nem mondható el. (A felhasználók szempontjából ezek a jogi problémák mellékesek, hiszen magánhasználatra a KDE alapját képező Qt függvénykönyvtár is ingyenes.)

Használható-e már a GNOME?

A GNOME jelenlegi állapota a „név nélküli 1.0-ás" verzióval írható le leginkább. Ez nem azt jelenti, hogy ez az első verzió, hanem ez a régen várt, első igazán említésre méltó változat a sok használható – néha stabil, néha teljesen labilis – fejlesztői változat után. Miguel a LinuxWorld Expón jelentette be hivatalosan az 1.0-s verzió megjelenését. Ez sokak szerint elhamarkodott lépés volt, hisz a megjelenéssel egy időben történt meg a menürendszer áttervezése, valamint sok fejlesztő (egyetemista lévén) a vizsgaidőszakból eszmélve ekkor kezdett el foglalkozni azokkal a részekkel, amelyeket addig írt. Mindezek ellenére ez a csomag egy átgondolt, széles körben alkalmazható grafikus rendszer. A fejlesztés persze nem állt meg, a megjelenés után pár nappal is negyvennél több változtatás történt a GNOME forráskönyvtár szerkezetében. A csomag elég sok programot tartalmaz, hogy csak a közismert mc-t, a Midnight Commandert vagy a GIMP-et említsük. Tehát akik hiányolnak valamit a csomagból, vagy keveslik a hazai nyelvre átírt alkalmazást, azok május körül kielégítő választ kaphatnak egy újabb, stabil változat megjelenésével.

Telepítés

A GNOME elérhető forráskódban, RedHat (rpm) és Debian (deb) csomagokban is a GNOME ftp-szerverén és tükrözésein egyaránt. A legutolsó fejlesztői változatot a GNOME anonymous CVS-én keresztül kérhetjük le. Itt most csak a forrásból történő telepítést is-

mertetem, mivel a disztribúciókhoz készült csomagok tartalmazzák a helyes telepítési sorrendhez szükséges információkat, és egy paranccsal telepíthetők.

A GNOME-ot több külön csomagban terjesztik. Ezeket meghatározott sorrendben kell feltenni. (Az FTP-szerveren és a CD-mellékleten található README file tartalmazza a megfelelő sorrendet.) A fordítás során az egyes csomagok helyigénye akár 80 MB is lehet. Ha az objectfile-okat nem takarítjuk el (a make clean parancs segítségével az installálás után), akkor a teljes forrásfa (az installált példánnyal együtt) akár a 600 MB-ot is elérheti. A GNOME a GNU automake/autoconf rendszert használja, ami lehetővé teszi számára, hogy a programok paramétereit a fordítás előtt pontosan a rendszerünkhöz igazítsa.

Egy csomagot a következő módon kell telepíteni:

- Belépünk a kicsomagolt forrás könyvtárába.
- Lefuttatjuk a ./configure --prefix=<ahova a programot installálni akarjuk> parancsot.

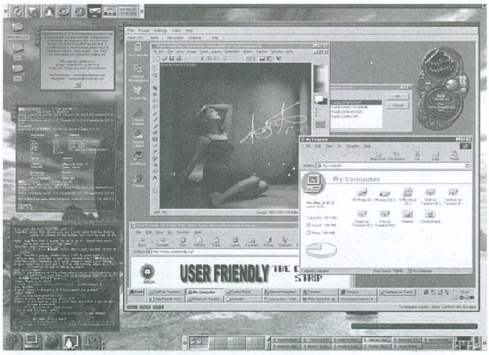
Javaslom, hogy ne az alapértelmezés szerinti „/usr/local"-ba telepítsük, hanem mondjuk a /usr/local/gnome alá. Így, ha új verziót teszünk fel, akkor elég csupán a könyvtárstruktúrát kitörölni, és máris „tiszta lappal" indulhatunk. Ha több GNOME vagy GTK-verzió van a gépünkön, akkor érdemes a PATH környezeti változót kiegészíteni a GNOME futtatható file-jainak könyvtárával (pl.: export PATH=/usr/local/gnome/bin:$PATH) Erre azért van szükség, mert a ./configure programok felhasználják az ebben a könyvtárban található config file-okat a segédkönyvtárak verzióinak és helyének meghatározására.

Szintén fontos a /etc/ld.so.conf file-ban a megfelelő könyvtárat beállítani (pl.: /usr/local/gnome/lib). Ha ez nem lehetséges, akkor az LD_LIBRARY_PATH környezeti változónak kell megfelelő értéket adni.

- Futtatjuk a make parancsot.
- Root jogosultsággal kiadjuk a make install, majd az ldconfig parancsot.
- A make clean parancs megszabadít az immár felesleges object file-októl sok helyet takarítva meg ezzel. A GNOME használatát a core csomag feltelepítése után kezdhetjük meg. Ha további csomagokat telepítünk, akkor azok automatikusan beépülnek a rendszerbe.

A GNOME használata

Ha sikerült feltelepítenünk a core csomagot, akkor a gnome-session program elindításával kezdhetjük meg a GNOME használatát. A felhasználónk könyvtárában létrehozhatunk egy .xinitrc (vagy .xsession, ha xdm-et



Egy GNOME és Enlightenment alapú munkafelület

használunk) nevű file-t, amelybe a következőket érdemes beírni:

exec kedvenc_ablakkezelőnk & gnome-session

Az előbbit csak akkor érdemes beírni, ha nem tetszik az alapértelmezett ablakkezelő, egyébként annak indítását is a GNOME-ra bízhatjuk.

Ekkor először az ablakkezelő indul el, majd a GNOME, így a „kilépés" gombra kattintva az X felületből is kiléphetünk.

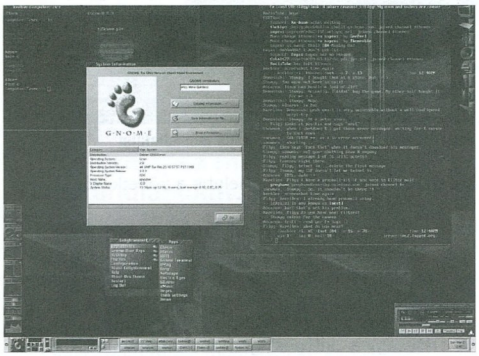
Ha belépünk megjelenik a panel, valamint a gnome-help-browser a képernyőn, és birtokba vehetjük új desktopunkat. A gnome-session program felelős azért, hogy minden indításkor ugyanazok a programok induljanak el, amelyek a kilépés előtt futottak. Ha az alkalmazás GNOME-kompatibilis, akkor pontosan abban az állapotban fogjuk találni, ahogy otthagytuk (pl.: ugyanaz a file lesz megnyitva a helpbrowserben). Ha a GNOME indítása előtt beállítjuk a LANG környezeti változót a hu_HU (vagy csak simán hu) értékre, akkor abban a kellemes meglepetésben lesz részünk, hogy azok a GNOME programok, amelyekhez elkészült a fordítás, magyarul szólnak hozzánk.

Barátkozás a panellel

A GNOME lelke a panel. Ez szolgál a programok gyors indítására, és a beágyazott alkalmazások (appletek) futtatására. A rendszer legtöbb funkciója a panel segítségével, egy kattintással elérhető. Tartalmaz egy főmenüt, amelybe az összes GNOME-kompatibilis alkalmazás regisztrálja magát. Alapértelmezésben megtalálható benne a GIMP és a Netscape indítója is. Saját magunk is definiálhatunk indítókat, illetve a menüből a leggyakrabban használt programjainkat egy-

szerűen húzd és ejtsd módszerrel) áthelyezhetjük közvetlenül a panelre.

Egyszerre több panelt is használhatunk. Megkülönböztetünk „él-" és „sarokpaneleket". Az élpanel a munkatér valamelyik szélén helyezkedik el, a rendelkezésére álló helyet teljesen kitöltve. A sarokpanel csupán annyi helyet foglal, amennyi a rajta található objektumok megjelenítéséhez szükséges. A panelt el is rejthetjük a két szélén található nyomógombokkal. Természetesen



Egy GNOME és Enlightenment alapú munkafelület

automatikus elrejtés is beállítható. A képernyő közepe felé mutató gomb a másik sarokba mozgatja a sarokpanelt. A középső egérgombot használva tudjuk a panelt áthelyezni a képernyő különböző sarkaiba.

Lehetőségünk van arra is, hogy a panelünkhöz „fiókokat" adjunk hozzá, ami leginkább egy menüre hasonlít, viszont appleteket és programindítókat helyezhetünk rá. Ez lehetővé teszi, hogy a rendelkezésre álló helyet megnöveljük. Az első indításkor a Help Browser, a Control Center, és a Netscape indítója jelenik meg a panelen.

A Help Browser (nevéhez híven) hozzáférést nyújt a gépünkön található dokumentációhoz. Képes kezelni a man és info oldalakat, valamint a HTML-file-okat. A GNOME programok dokumentációja leggyakrabban html formátumban találhatók. Mivel a Help Browser ismeri a http protokollt, ezért egyszerű webböngészőként is használható. A panelen futó alkalmazásokat pedig appleteknek hívjuk, ezek kisebb programok vagy programkönyvtárak, melyek közvetlen kapcsolatban állnak a GNOME panellel és magával a környezettel, nélküle azonban életképtelenek. Indításuk a főmenüben elhelyezett panel almenü, applet hozzáadása menüpontjában lehetséges. Az appletek törlésükig (csak a futást töröljük, a program megmarad)

Szórakoztatás

- Hal – Egy hal, amelyre kattintva a fortune program üzeneteit kapjuk meg animáció kíséretében. Egy nagyon egyszerű tamagochiként is felfogható. Fejlesztői szerint csupán idő és lemezterület po-csékolás.
- Tizenöt – Tizenötös puzzle-játék a lehető legkisebb méretben.
- Slashapp – A slashdot.org híranyagának legújabb címeit mutatja.

Monitorok

- Egységes arculatú akkumulátortöltöttségjelző, valamint processzor-, memória-, lemezterület- és swaphasználat-figyelő programok.

Multimédia

- CD-lejátszó – Miniatűr kezelőfelület CD-lejátszáshoz, nagyon ötletes.
- Mixer – Hangerőszabályzó.

Hálózat

- Tárcsázó, levélfigyelő, modem LED emuláció, web irányítópult, valamint hálózati forgalommérő applet.

Segédprogramok

- AfterStep óra – AfterStepből ismerős falinaptárral kombinált óra.
- Karaktertábla – speciális karaktereket tehetünk ki rá, ami a billentyűzetről hiányzik.
- Óra.
- Levélfigyelő és óra.
- Lemezbeillesztő (drive mount) – ZIP-drive, CD-ROM, floppy, cserélhető merevlemez gombnyomással való beillesztését (mount) teszi lehetővé, jelzi az eszköz státuszát.
- GNOME Pager – virtuális lapkezelő, GNOME-kompatibilis ablakkezelőkhöz.
- Mini Commander – Egy parancsértelmező makrózási lehetőségekkel, history-val, parancs-kiegészítéssel.
- Nyomtató applet – Nyomtatóikon, amelyre rá lehet húzni GNOME-os programokból dokumentumokat a nyomtatás megkönnyítésére.
- GKB billentyűzet-átkapcsoló – Billentyűzettábla (keymap) kapcsoló program, zászlóval jelzi az aktuális billentyűzetkiosztást, erre kattintva lehet váltani. Érdekessége, hogy magyar és amerikai a két alapértelmezett billentyűzet.

futnak, addig minden indításkor rátöltődnek a panelra. A keretezett részben látható jó néhány applet felsorolása és rövid leírása.

A GNOME beállításai

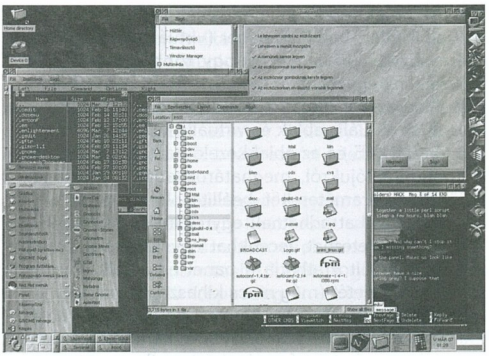
A GNOME beállításait legegyszerűbben a Control Centerben (Vezérlőközpont) végezhetjük el. A megjelenő ablak két részre van osztva. A bal oldalon láthatjuk a konfigurációs programok listáját, feladatuk szerint fastruktúrába rendezve. Jobb oldalon mindig az éppen aktív konfiguráló program képét látjuk. Így gyorsan és egyszerűen érhetjük el a GNOME desktop összes beállítását. Ezeket a kis programokat angol szóval cappletnek nevezik. Az elérhető funkciók természetesen attól függnek, hogy milyen programokat installáltunk. Alapállapotban a következő funkciók érhetők el.

- Lapkezelő – Ebben a menüben a munkaasztal beállításait, vagyis a háttér, a képernyővédő és az ablakkezelő tulajdonságait állíthatjuk be, valamint a témaválasztás is ide került be, amivel futás közben meg lehet változtatni a GNOME-os programok kinézetét.
- Multimédia – A GNOME hangrendszerének beállításai.
- Peripherials – Itt a beviteli eszközök tulajdonságait adhatjuk meg.
- Startup programs – A bejelentkezéskor elindítandó programokat állíthatjuk be.
- Szövegszerkesztő – Melyik szövegszerkesztőt indítsa a GNOME alapértelmezésben, ha valamilyen file-t kell szerkeszteni.
- URL-kezelők – A súgó és egyéb file-okhoz rendelhetünk alkalmazásokat.
- User interface options – A gtk és a GNOME alapjellemzőit – az MDI- és párbeszédablakok – egyszóval az alkalmazások megjelenését állíthatjuk be egységesen.

Néhány műveletet a drag and drop módszerrel is elvégezhetünk: ha kiválasztunk egy színt, majd a panelre dobjuk, akkor a panel színe a megadottra változik (ez az ötlet az OS/2-ből lehet ismerős – a szerk.).

File-kezelő

A GNOME alapértelmezésben a Midnight Commandert használja file-kezelőként. Ez meglepőnek tűnhet, hiszen az mc-t legtöbben karakteres felületű Norton Commander-klónként ismerik. Az elmúlt hónapokban viszont elkészült a grafikus felülete is. Fontos megemlíteni, hogy az mc VFS-én (Virtual File System) alapul a GNOME hasonló rendszere. Ezáltal elérhető, hogy a szabványos



A GNOME file-kezelője: a GNU Midnight Commander

file-rendszereken kívül az FTP és HTTP protokollok, tömörített állományok, mcfs (az mc saját hálózatos file-rendszere), illetve az undelfs (az ext2 file-rendszeren törölt állományok visszaállítását segítő file-rendszer) egységesen elérhetőek legyenek. A későbbiekben tervezik, hogy a GNOME konfigurációs file-jai is elérhetők lesznek a file-kezelőből egy hasonló rendszeren keresztül.

A GNOME rendszerben (a KDE-hez hasonlóan) filekezelő menedzseli a desktopon megjelenő ikonokat. Egyszerűen a munkaasztalra helyezhetünk akár egy könyvtárat, akár egy file-t a gyorsabb elérés érdekében. A file-kezelő használata szinte semmiben sem különbözik a más rendszerekben megszokottaktól. Az egér segítségével jelölhetünk ki, másolhatunk, mozgathatunk file-okat, könyvtárakat. A GNOME a háromgombos egerek lehetőségeit is kihasználja. Például, ha a file-kezelőben a középső gombbal rákattintunk egy könyvtárra, akkor új ablakot nyit, amiben a megjeleníti a könyvtár tartalmát.

Ablakkezelők

A GNOME-hoz nem tartozik egy bizonyos ablakkezelő (window manager), mint a KDE-hez. Olyan ablakkezelőt használhatunk, amilyet szeretnénk. Mindazonáltal vannak olyan funkciók, amelyek csak GNOME-kompatibilis ablakkezelővel érhetők el. Jelenleg egyedül az Enlightenment támogatja teljes mértékben a GNOME-ot, így a fejlesztők ezt használják referenciaként. A következő funkciókat biztosítja egy GNOME-kompatibilis ablakkezelő. Ezek miatt érdemes használni:

- A GNOME közzéteszi bizonyos objektumait, például a menüit az ablakkezelő számára, így azokat be lehet illeszteni az onscreen menübe.

● Az ablakkezelő minden ablakról információt tart fent a GNOME-ban lévő pager (virtuális lapkezelő) számára. Ez lehetővé teszi, hogy több munkaterületet használjunk egyszerre.

● Mind az ablakkezelő, mind pedig a GNOME tudja, melyik az aktuális ablak és virtuális lap.

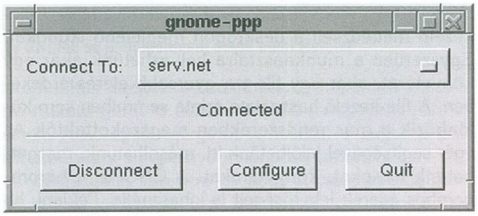
● A GNOME és az ablakkezelő adatokat tud cserélni a konfigurációjukról, meghatározhatják egymásnak a különböző paramétereket, beállításokat.

● Parancsokat adhatnak egymásnak, a GNOME bezárhat, átméretezhet, lecsukhat egy ablakot, az ablakkezelő megváltoztathatja a panel és az alkalmazások témáját, kinézetét (még nincs kihasználva teljesen).

● Az ablakkezelő nem keretezi be azokat az ablakokat, paneleket, amelyeket GNOME nem akar, valamint az ablaktípusokat egyformán értelmezik.

A GNOME hangjai

Unix-rendszereken a hanglejátszás (hasonlóan bármely másik hardvereszköz eléréséhez) egy speciális eszközfile-on (általában a /dev/dsp-n) keresztül történik.



A GNOME PPP tárcsázója

Egyszerre csak egy program használhatja ezt a file-t, viszont sokszor előfordul, hogy több programnak a hangját szeretnénk hallani.

Ezért a GNOME egy hangvezérlő programot használ, az Enlightenment Sound Daemont (ESD). Ennek célja, hogy a különböző forrásokból származó hangokat összekeverje, és egy egységes hangként továbbítsa a /dev/dsp-nek. Természetesen a régi – közvetlenül az eszközfile-t használó – programok is futtathatók vele egy speciális alkalmazás (esddsp) segítségével, mely a /dev/dsp-nek szóló hívásokat átirányítja az ESD-nek.

A GNOME Control Centerben különböző eseményekhez hangokat definiálhatunk, így a számítógép bármely ténykedésünkre hanghatással felelhet. A program tervezőit dicséri, hogy a funkció egyetlen kattintással kikapcsolható.

Hasznos programok

Már most is sok segédprogram létezik GNOME-hoz. A következőkben ismertetünk közülük néhányat.

● Gnome Terminál – Nevéhez híven egy terminálprogram, hasonló a jól ismert xtermhez. Érdekessége, hogy képes a hátterét egy megadott képfile-ra változtatni, így munka közben is kedvenc háziállatunk képét szemlélhetjük.

● Session Manager Properties – Ennek a programnak a segítségével megadhatjuk, hogy mely programok induljanak el automatikusan a GNOME indulásakor.

● GNOME Menüszerkesztő – Megkönnyíti a GNOME menüinek szerkesztését. Használata egyszerű és kényelmes.

● Electric Eyes – A GNOME saját képnézegetője. Gyors, viszonylag sok formátumot kezel, és néhány egyszerűbb képszerkesztési műveletre is képes. Kezel képlistákat, amivel sok kép végignézése kényelmesen megoldható.

● Gnome Kereső – Ha valaki nem kedveli a find és a grep program szintaxisát, akkor javaslom, próbálja ki ezt a programot, ami egy jól konfigurálható grafikus felületet biztosít a két remek segédeszközhöz.

● Log Viewer – Unix endszereken fontos hogy mindennap elolvassuk a naplófile-okat (log-okat), amit a szorgalmas démonok gyűjtöttek össze a napi munka során. Biztonsági szempontból is fontos a rendszernapló folyamatos követése, hiszen ide kerülnek (alapértelmezésben) azok a bejegyzések, amelyek gépünk elleni támadásra utalhatnak. Erre a feladatra készült a logview program, amivel kulturált környezetben, napokra bontva szemlélhetjük gépünk üzeneteit.

● Felhasználólista – A w nevű unixos programnak is elkészült a GNOME alapú változata. Segítségével nem csupán a bejelentkezett felhasználók adatait tekinthetjük meg, hanem programokat is futtathatunk. Így miután ellenőriztük, hogy az ismerősünk belépett, egy kattintással indíthatjuk a Talk programot, és máris kezdhetjük a csevegést.

● Gmix – Keverőpult OSS eszközökhöz. Az aumix GNOME-os átiratának nevezhetnénk leginkább, kilépéskor a beállítást elmenti.

● Gcd – GNOME CD-lejátszó CD-adatbázissal.

● Gcalc –Tudományos számológép nagy kijelzővel. Hasonlít az xcalc-hoz.

● GNOMECard – Elektronikus névjegykártya adatbázis. Ismerőseink adatait (név, születésnap, munkahely, beosztás, e-mail, telefonszám, időzóna, földrajzi elhelyezkedés) tárolhatjuk benne. Természetesen kereshetünk az így nyert névjegy-adatbázisban.

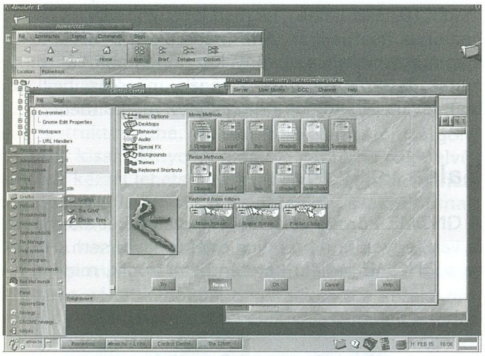
● GNOMECal – Naptár, illetve nemcsak naptár, ha-

nem ütemező és határidőnapló is, a leghasznosabb alkalmazás egy elfoglalt ember számára. Napi, heti, hónapos és éves bontásban láthatjuk időnket. Segít eligazodni a dátumok, naptárlapok, időpontok dzsungelében. Adott időpontokhoz eseményeket rendelhetünk, akár ciklikusan bekövetkező eseményeket jelölhetünk saját naptárunkba. Az események bekövetkezése előtt kérhetünk figyelmeztetést: mely megjelenő figyelmeztető szöveg, hang, induló program, vagy emlékeztető levél lehet. Kivételes eseteket adhatunk meg, megkereshetjük egy esemény időpontját vagy az időpontra előjegyzett történéseket. A program tetszetős, szinte mindent be lehet állítani, a kezelése egyszerű.

● GNOME névjegy – Rendszerinformációk GNOME módra. Egy panelen megjelennek a rendszer főbb paraméterei, valamint egy kis reklám a GNOME fejlesztőkről. A részletes információk gombra kattintva megjelennek a memória, lemezterület, valamint processzorhasználtság pillanatnyi értékei.

Hogyan működik a GNOME?

A GNOME jelenleg az ORBit nevű CORBA-kompatibilis Object Request Brokerre épül, melynek feladata a különböző komponensek, alkalmazások közötti kommunikáció lebonyolítása.



GNOME asztal és vezérlőpult

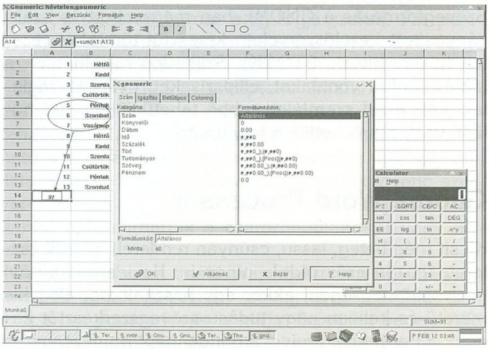
A programokat a gnome-session fogja össze, indítja el, valamint a történéseket a beállítások kezelője jegyzi. Minden egyes alkalmazás szabványos konfigurációs eljárással rendelkezik, így egy helyen tárolódnak jellegzetesen a felhasználó .gnome könyvtárában. A belső felépítés objektumorientált és a CORBA-nak köszönhetően programnyelv-független.

A panel csak egy a sok alkalmazásból, amely elindul egy gnome-session indításakor, hiszen minden applet, amelyet a panelra tettünk külön futó program, és csak összeköttetésben áll a panellal. Ezen kívül az ORBit kommunikációért felelős része is fut, és indításkor control center részei, az egér, billentyűzet, háttér és más környezeti elemek beállításáért felelős programok is lefutnak.

Felsőbb utasításra mindegyik program újra tud indulni, megáll, újraértelmezi a beállításait tartalmazó állományokat, vagy egyszerűen csak frissíti önmagát. Mindez csak egy utasításba kerül, a gtk+ programkönyvtár és a glib képességeinek köszönhetően. A GNOME erőssége szintén a gtk+ miatt a „témázhatóság", és a témák futás közbeni változtatása. A GNOME-kompatibilis ablakkezelőkkel való kommunikáció is ide tartozik, hiszen egy futó Enlightenment ablakkezelő és egy control panel meg tudja beszélni, mik az ablakkezelő aktuális beállításai, ezek mire változzanak. Ezután az ablakkezelő mindent teljesít a GNOME kérésének megfelelően, anélkül, hogy ki kellene szállni egy alkalmazásból is.

Alkalmazások

Egyetlen desktop-rendszer sem lehet sikeres, ha nem léteznek hozzá megfelelő alkalmazói programok. Lássuk, a GNOME hogyan felel meg ennek a követelménynek! A GNOME alkalmazásokról jelenleg az mondható el, hogy sok ígéretes kezdemény létezik, de kevés a stabil, jól használható szoftver. A következőkben néhány figyelemre méltó programot ismertetünk, melyekre érdemes lesz odafigyelni az elkövetkezendő hónapokban.



A Gnumeric, a GNOME táblázatkezelője

Gnumeric – Excel for Linux?

Az Excel-stílusú és -tudású táblázatkezelő programokkal nem igazán lehet Dunát rekeszteni Linux alatt. A GNOME csapat készít egy táblázatkezelő programot ennek az igénynek a kielégítésére, ami az egyik ütőkártyája lesz a GNOME programcsomagnak.

Tudása az Excel képességeihez mérve szerény, eddig 58 függvényt írtak meg, köztük a matematikai, boole algebra, statisztika és valószínűség számítás, valamint konverziós és konstansokat visszaadó (pl.: PI) valamint az egyéb speciális (dátum, időpont, a kijelölt területet visszaadó) függvényeket találhatjuk meg.

A programhoz kiegészítéseket (plug-ineket) is írnak, és írhatunk mi is, ezeket hozzáilleszthetjük a Gnumerichez, akár működés közben is. Az Excelben megszokott automatikus kitöltés is megtalálható, tehát ha két szomszédos cellába beírunk egy egyest és egy hármast, akkor a kettőt kijelölve, a cella jobb alsó sarkában lévő pontot megfogva és meghúzva a megfelelő irányba, a sorozat folytatódni fog (5,7,9...). Ugyanez működik a hét napjaival is, tehát ha egy cellába beírjuk, hogy Hétfő olyan cellasort kapunk, ahol a hét napjai a megfelelő sorrendben lesznek felsorolva.

Hiányosságai között szerepel a grafikonkészítés lehetősége, valamint a makrózás. Ezek várhatóan elkészülnek a végleges verzió megjelenéséig. Mentés jelenleg XML (HTML-stílusú általános leíró nyelv, a GNOME ezt használja – az XML egy SGML megvalósítás... a szerk.) formátumban lehetséges, betöltés pedig XML, valamint a CSV (az Excel által elmentett, az oszlopokat vesszőkkel elválasztó file) formátumból, valamint bizonyos .XLS file-okból (az egyszerűbbekre célzok itt, amelyekben kevés képlet található. Pozitívum, hogy a file tartalmazhat több munkalapot is.) Szövegformázás a GNOME-ban megszokott és szabványosított kezelőszerveken keresztül történhet (ezeket itt úgy hívják, STOCK). Szabadon rajzolhatunk a táblázatunkra vonalakat, ellipsziseket, nyilakat, téglalapokat, ezzel szemléletesebbé tehetünk egy számolást, vagy tükrözhetjük a folyamatot.

Gnome Word Processor

Kicsit még beteg, lassú, csúnyán rendereli a betűkészletet, igazán erős oldala nincs is. Ha viszont kinövi gyermekbetegségeit, akkor egy használható, elég kis méretű és kellően nagy tudású szövegszerkesztőt kapunk a kezünkbe, amivel a mindennapi munkát, gyors formázást igénylő dokumentumokat, leveleket, meg tudjuk írni. Jelenleg is tudja azokat a funkciókat, melyek szövegszerkesztővé teszik. Betűtípust, írásmódot és méretet állítani, szöveget pozicionálni, színezni, képet beszúrni, többszintű felsorolásokat kezelni. Táblázatkezelése új alapokra lett helyezve. Elfelejthetjük, hogy van kerete a táblázatnak, ugyanis léteznek a cellák, és kész. Minden cella után jön egy másik cella, természetesen a sorban vagy oszlopban utolsó után nem. Ami ezután meghatározó egy cella méretében, az a benne elhelyezkedő dokumentumrészlet, tehát szöveg, kép, valamint táblázat. Első ránézésre kicsit bizarr, de használható, sőt pár perc használat után kifejezetten tetszik. Természetesen helyesírás-ellenőrzés is van benne, bár ez unixos, ispell alapú, ami alá még mindig nincs megírva a magyar nyelvű támogatás. Van még egy kicsi, de aranyos kiegészítés: szószámlálás (word count), ami egy párbeszédablakban kidobja, amit egy unixos wc mondana a szövegről. Természetesen itt is épül a plug-in-rendszer, sok-sok file-formátum és varázslat fog ezáltal belekerülni a GWP-be.

A Beállítások (Properties) panel szegényes még, egyenlőre az alapértelmezett színek beállítására szolgál, valamint be lehet állítani a vonalzóskála-egységeket (cm vagy coll), ki-be lehet kapcsolni a vonalzókat. Továbbá be lehet kapcsolni egy gyorsrajzolás-módot, ami gyors, viszont sokat kell alkalmazni a Nézet menüben lévő frissítés menüpontot, mert egy idő után teleszemeteli a lapot. Be lehet állítani még a debug-módot (fejlesztőknek), és meg lehet engedni az összes betűtípus használatát. Ami megszokott a GNOME-os alkalmazásoknál: szabvány kezelőgombok (új, megnyitás, bezárás, mentés, kivágás, másolás, beillesztés), leválasztható, ide-oda rakható, mint a menü.

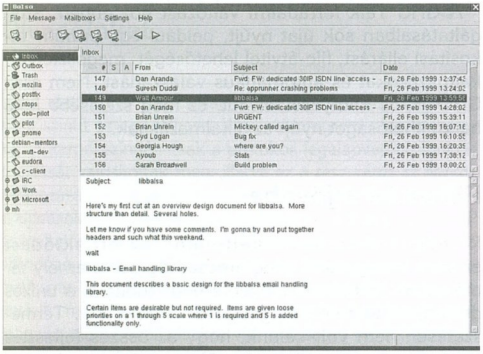
Balsa vagy M?

A GNOME környezet alatti levelezésre két alternatíva is van, egyelőre nem teljes értékű egyik sem. A Balsa a GNOME projekt keretén belül íródik, minden bi-

A Balsa logója

zonnyal kicsit jobban fog illeszkedni a kezelőfelülethez. A GNOME-környezet adta lehetőségekből már a fejlesztési stádiumban is sokat kihasznál, kezelhető,

bár nem minden szempontból kielégítő levelezőprogram. Képes távoli szerverekről IMAP és POP3 szolgáltatáson keresztül leszedni a leveleket, valamint távoli szervert használni levélküldésre. Szűrőket tehetünk az alkalmazásba, ami különböző mappákba válogatja keresési feltételek teljesülése alapján a leveleket. Az M is rendelkezik ezekkel a képességekkel, sőt van címtár (address book), lehet Python programnyelven kisebb kiegészítéseket írni hozzá, külső szerkesztőprogram használatát (ez a GNOME-ban felesleges, be lehet állítani az alapértelmezett szövegszerkesztőt) is engedi, valamint hírcsoport (news) olva-



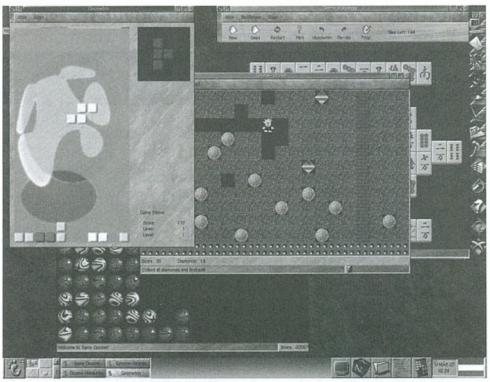
Balsa, a GNOME levelezőprogramja

sóként is használható. Az M címtár szolgáltatása nem elég egyértelműen kezelhető: ha nem használtuk még, hamar belezavarodhatunk a csoportok és egyének fastruktúrájába. Kezelői felülete viszont átgondolt, bár kissé hiányos. Például a címtárba való felvétel csak kézzel lehetséges, ami ráadásul kissé nehézkes. A csatolt file-ok lementésével is akadhatnak problémáink.

Ami a két programban igazán elragadó, az a törlések színpompás változatossága. Behúzhatjuk a levelet a kukába, rákattinthatunk a kuka ikonra a kezelőszerveknél, jobb gombos menüben kiválaszthatjuk a törlés menüpontot, esetleg a főmenü megfelelő menüjéből kiválaszthatjuk ugyanezen pontot vagy d billentyűt nyomhatunk a törlendő levélre.

Hogyan üssük el az időt?

A gnome-games csomag remek lehetőséget nyújt a kikapcsolódásra. Több mint tíz játék közül választhatunk, melyek között olyan örökzöld példányok találhatók, mint a Tetris, Aknakereső Mahjongg, Freecell. Elkészült a Robots nevű unixos játék GNOME-verziója is, így az eredetileg szöveges játékot továbbfejlesztve,



GNOME-hoz írt játékok

grafikus környezetben élvezhetjük. A Gyahtzee segítségével, a kockákkal is kipróbálhatjuk a szerencsénket. Azoknak, akiket ezek a logikai játékok nem hoznak lázba jó hír, hogy készül a GNOOM, a DOOM GNOME-osított változata. Ennek ellenére az a véleményem, hogy a GNOME platform inkább a logikai és stratégiai játékoknak kedvez, ahol nem a tiszta, gyors mozgás a legfontosabb.

Kommunikáció, magasabb szinten

XChat, irssi, yagirc – Néhány IRC-kliens

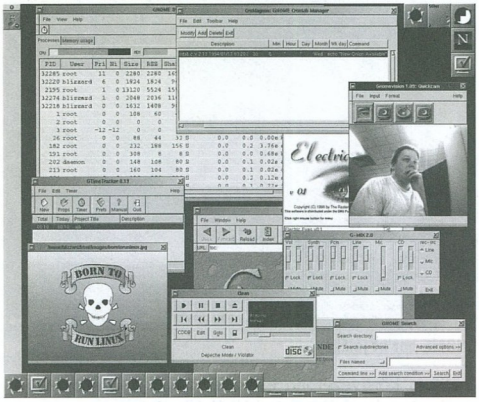
Aki sokat használt grafikus IRC-klienseket, az az XChat programot hamar meg fogja szokni. Pár percnyi beállítás után olyan otthonossá lehet tenni, hogy szinte minden igénynek megfelel.

Erőssége, hogy Perl szkripteket írhatunk hozzá, valamint a GNOME-nál már-már megszokott plug-in-rendszert is támogatja, tehát bármikor megsokszorozható a szolgáltatások száma. Induláskor egy szerverlista ablakot kapunk, ahova beírhatjuk kedvenc szerver–csatorna párosunkat, beállíthatjuk nickjeinket, valamint megnyithatjuk a csatornaablakokat vagy noteszlapokat.

A csatornaablak beállításai között van egy nagyon hasznos opció: az üzenet, szöveg elhangzásának másodpercre pontos idejét is kiírja. Színezett becenevek, becenév-kiegészítés és hibajavítás, csatornaoperátori funkciók megkönnyítése, felhasználólistán jobb gom-

bos menüben lekérdezések, ilyen és sok hasonló, hasznos apróságot találunk a programban.

Mellesleg jegyzi a beszélgetés között elhangzó URL-eket, ezeket az éppen futó Netscape-ben meg lehet jeleníttetni. A DCC küldésre és fogadásra, valamint DCC beszélgetésre külön ablak szolgál. Az irssi rendelkezik mindezekkel a képességekkel, kivéve az időpont megjelenítését, viszont a panelra föltesz egy appletet, amin megjelenik az aktív csatornák neve, az egeret efölé mozgatva megjelenik a csatorna forgalma, akkor is, ha épp nincs nyitva a csatorna ablaka. Emellett figyeli a beszélgetést, és más színnel jelöli a nekünk szóló üzeneteket.



Egy GNOME és WindowMaker alapú munkafelület

A yagirc, mint neve is mutatja (Yet Another GTK+ IRC) már csak a választási lehetőségek bővítésére szolgál. Amivel többet nyújt: háttérkép csatornaablakba tétele, játékos, könnyed kinézet. Emellett persze az alapszolgáltatások is megtalálhatók, és a jövőre nézve elég komoly elképzelése van a fejlesztőgárdának.

GtkICQ

Egy teljes értékű ICQ-kliens. Csupán a file-átvitel képességét hiányolhatjuk, de már nem sokáig. Minden mást tud, amit eddig tudnia kell. A panelen elhelyezkedő applet jelzi az állapotát, valamint a szokványos ICQ főablak jelenik meg, ahol a szolgáltatásokat elérhetjük, ismerőseinket bejegyezhetjük. Első indításkor kérelmez nekünk egy Univerzális Internet Számot, ha még nincs, valamint feljelentkeztet egy általa ismert ICQ szerverre. Ezzel levette a vállunkról az adminisztráció gondját, már csak meg kell keresnünk barátainkat, amit megtehetünk egy keresőablak segítségével. Használata egyszerű, azoknak is, akik nem használtak előtte semmilyen platformon ICQ-t.

GNU Talk

Egy Talk-kliens és démon, amely új alapokra helyezi a Talkot. A Talk a Unix-rendszerek online (karakter karakter után) beszélgetőrendszere volt. Első változatai két felhasználó közvetlen kapcsolatát engedték meg. Később megírták az ytalk programot egy nagyobb programcsomag részeként, amely már több felhasználó beszélgetését is lehetővé tette, de nem terjedt el igazán a napi használatban.

A GNU Talk forradalmi változást hoz ebben. Szolgáltatásaiban sok újat nyújt, például egy primitív titkosítási eljárást, file-átvitel lehetőségét, megosztott alkalmazásokat és automatikus válaszadást. Nem csak GNOME-os környezetben fut, ez még szélesebb körű használhatóságot nyújt az alkalmazásnak.

Pillantás a jövőbe

Reméljük sikerült felkeltenünk az érdeklődést ez iránt a nagyszerű programcsomag iránt, amely reményeink szerint az elkövetkezendő időkben a unixos desktop-rendszerek meghatározó eleme lesz. Természetesen nem volt célunk, hogy az összes fejlesztés alatt álló programot ismertessük, csupán a rendszerről kívántunk áttekintést adni. Éppen ezért érdemes rendszeresen ellátogatni a http://www.gnome.org weblapra, ahol a legfrissebb információkon kívül az összes GNOME-kompatibilis program listáját is megtalálhatjuk.

A frissen megjelent GNOME alapra épült programok között egyre több a hasznos, sőt hiánypótló alkalmazás. A több mint 200 fős, a világ minden táján szétszórt fejlesztőcsapat fő célja a rendszer minél inkább felhasználóbaráttá tétele. Ez látszik is az eredményen, a kezdő Linux felhasználók nem kis örömére. ■

MUTATÓ

Hasznos webcímek GNOME-mal kapcsolatban:
A GNOME honlapja: a http://www.gnome.org
A GTK, (melyre a GNOME is épül) weblapja: http://www.gtk.org
Témák a GTK-hoz: http://gtk.themes.org
Az Enlightenment ablakkezelő weblapja: http://www.enlightenment.org/
Témák az Enlightenmenthez: http://e.themes.org

Slapic
slapic@vogel.hu

Netre fel!

Többször írtunk már arról, hogyan lehet modemes Internet-kapcsolatot létrehozni Linux alatt, így most elsősorban a CD-n megtalálható Debian GNU/Linux részleteivel, és néhány hasznos trükkel ismertetnénk meg Olvasóinkat.

Linux alatt a PPP-kapcsolat két szinten valósul meg. Egy kernelszintű és egy felhasználószintű program végzi a munkát. A kernel szintjén csak a protokoll legalapvetőbb funkcionalitását valósították meg, illetve ez a rész végzi az eszköz létrehozását, az útválasztást, valamint a különböző tömörítési eljárásokat. Ahhoz, hogy ez működjön, szükség van a PPP szolgáltatásra, amit modulba érdemes tenni, hiszen nincs rá állandóan szükség. A ppp-modul betöltését a kernel automatikusan elvégzi, mivel azon alapvető szolgáltatások közé tartozik, melyről pontos ismeretei vannak. Modulba fordításához az alábbi beállításokra van szükség a kernel konfigurációjában: CONFIG_PPP=m. A többi funkció, melyek a tömörítést végzik, automatikusan kerülnek fordításra a PPP mellé.

Modulként a /lib/modules/<kernel-verzió>/net/ könyvtárban találjuk őket, ppp_deflate és bsd_comp néven, de velük sem kell külön foglalkozni.

A felhasználóiszintű pppd program

Függetlenül attól, hogy milyen kapcsolat-felépítő segédprogramot használunk, legyen az a kppp, xisp vagy más, a kapcsolat tényleges kezelését a felhasználói szinten a pppd program fogja végezni. Jelenleg ez a Debian 2.1-ben a 2.3.5-ös verzió, mely már ismeri a 2.2-es Kernel ppp-funkciója által nyújtott összes szolgáltatást, mint például a feltételes kapcsolatot, de erről majd később. A pppd több konfigurációs file-t is használ, egyik sem kötelező, de érdemes megfontolni használatukat. Egy kivétellel az összes beállítás a /etc/ppp könyvtár alatt található, és ez a kivétel Debian-specifikus.

A /etc/ppp/options file-ban találhatók az általános beállítások, melyeket vagy ott, vagy a pppd parancssorában adhatunk meg. Ide érdemes írni minden olyan opciót, mely általános érvényű a rendszerre, vagy az esetek nagy részében igaz. Mivel ezt parancssorból vagy az eszközspecifikus beállításoknál felülbírálhatjuk, az sem okoz gondot, ha valahol mégsem ez kell. Az itt használható opciók listáját és leírását a man pppd parancs által kiírt dokumentáció írja le. Mint említettük, eszközspecifikus opciók is megadhatók. Ezek az options.eszköznév file-ba kerülnek, ahol az eszköznév a soros port neve Linux alatt, azaz ha a COM2-t használjuk ttyS1. A /etc/ppp könyvtárban találunk egy peers alkönyvtárat, melybe a szolgáltatóspecifikus beállításokat tehetjük, valamint egy /etc/chatscripts/ könyvtárat amibe a hozzájuk tartozó chat-szkriptek kerülnek. Ez utóbbira pontos hivatkozás az előbbi konfigurációs file-ban van, és ez végzi a modem kezelését, azaz az inicializálást, tárcsázást, illetve szükség esetén a szkriptes azonosítást. Ha PAP-azonosítást használunk, az ehhez szükséges azonosító és jelszó a /etc/ppp/papsecrets file-ba, ha CHAP-azonosítást használunk a chapsecrets file-ba kerül, illetve oda kell beírni. Szintén a /etc/ppp alatt találunk egy ip-up és egy ip-down szkriptet, valamint egy ip-up.d és egy ip-down.d könyvtárat. Ezeket használhatjuk programok futtatására, amikor a kapcsolat felépül, illetve lebomlik, de erről még lesz szó.

Beállítás

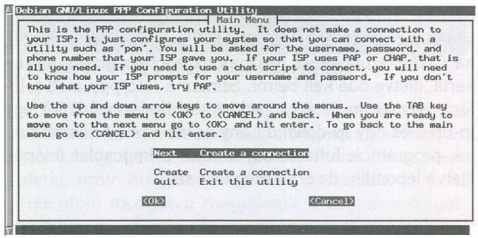
A pppconfig program

Debian Linux alatt találunk egy pppconfig nevű programot, amely a fenti állományokba beírja az általunk menüs rendszerben megadott adatokat. Az alábbiakban röviden ismertetjük a program kezelését.

A pppconfig elindítása után egy egyszerű menübe kerülünk, melyből a Next kiválasztásával léphetünk tovább, és kezdhetjük el létrehozni a beállítást. Az első dolgunk azt kiválasztani, hogy milyen azonosítást akarunk használni. Három választási lehetőségünk van: PAP, Chat vagy CHAP. Az első és az utolsó az ismert azonosítási eljárások, a második pedig a szkriptes módszert jelenti, azaz amikor login: vagy username: prompt, majd password: prompt után kell beírni az adatokat. Az ezután következőkben a választásunktól függően más és más kérdések következnek. A leggyakoribb, azaz a PAP-azonosítás alapján megyünk végig, és a végén teszünk említést a chat változatról. A CHAP beállítása a PAP-éval azonos módon történik.

A következő lépés a jelszó megadása, majd az azonosítónkat kell beírni. Ha ez is megvan, ki kell választanunk a megfelelő soros portot. Itt csak a már beállított portok közül választhatunk, azaz előbb be kell állítani őket a setserial paranccsal. A legáltalánosabb a /dev/ttyS1, azaz a COM2. A következő lépésben meghatározhatjuk, hogy az alapértelmezett átjáró ezen a kapcsolaton keresztül legyen elérhető. Ha ez az egyetlen kijáratunk az Internet felé, biztosan szükségünk van erre, azaz a „defaultroute" opciót válasszuk. Ha nincs rá szükség, akkor a „-defaultroute"-ot Az opciók között itt is és máshol is a TAB billentyűvel mozoghatunk, továbblépni az ENTER-rel lehet, ha az OK gomb az aktív; míg a CANCEL gomb „megnyomásával" kilép.

Az IP-szám meghatározása a soron következő feladat. Ha IP-számunkat dinamikusan kapjuk a szolgáltatótól, itt nem kell semmit megadni (kivéve, ha szeretnénk használni a pppd feltételes tárcsázási módját – amiről még lesz szó). Sok esetben csak a noipdefault opció megadása esetén fog működni a kapcsolódás, mivel előfordul

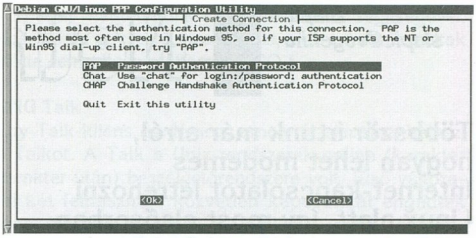


A pppconfig főmenüje

néhány rossz PPP-szerverprogram, amely ha a kliens felajánlja, hogy használják a rajta megadott IP-számokat, azt elfogadja, de adatot már nem továbbít felé.

Az IP-számot sajátszám:másik-gép-száma formában kell megadni, ahol bármelyik szám elhagyható, azaz a kettőspont előtt és mögött sem kell számnak állnia. Ha egyik számot sem adjuk meg, a kettőspontot sem írhatjuk ki. A mezőt üresen is lehet hagyni.

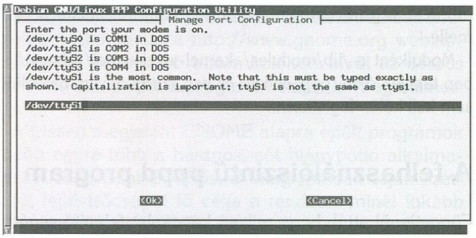
A modem és a gép közötti kommunikációs sebesség beállítása következik. Amennyiben nincs vele probléma, javasoljuk a legmagasabb sebesség, azaz átlagos soros port esetén a 115200 használatát. Fontos, hogy ez nagyobb legyen a modem névleges sebességénél, annak érdekében hogy az esetleges tömörített forgalom, melyet a modem a túlsó modemmel folytat, gond nélkül eljusson a gépig. Ezután a modeminicializáló-parancs következik. Ha a modem alapbeállítása jó, akkor ide egyszerűen ATZ-t kell írni. Javasoljuk, hogy a modemet mindenki egyszer állítsa be egy terminálprogram (például



Szabadon válogathatunk a PPP azonosítási módjai közül

minicom) segítségével, majd a beállításokat az AT&W paranccsal mentse el a modem nem felejtő memóriájába, így a továbbiakban ezzel nincs gond, és minden program számára általános beállításként szolgál. Ha a PPP-kapcsolatunkhoz egyedi modembeállító parancsokat kell vagy szeretnénk használni, azt adjuk itt meg.

A következő lépés a telefonszám megadása. Pontosan úgy adjuk meg a telefonszámot, ahogy azt a kapcsolat felépítéséhez tárcsázni kell. A tárcsázásban a szünetet a vessző jelenti (alapértelmezés szerint 2 másodperc), a vonalhangra a W beírásával várhatunk (az első vonalhangra mindenképpen vár a modem, ha ezt nem változtattuk meg). Távhívás (kék szám, Internetszám) hívásakor a legtöbb helyen nincs szükség arra, hogy a 06 után vonalhangra várjunk, egybe lehet írni a teljes számot. Fontos az is, hogy se kötőjelet, se szóközt ne hagyjunk a

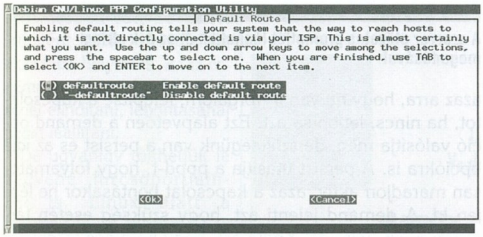


Sorban a soros port beállítása következik

számban. Ha mindez megvan, ki kell választanunk a tárcsázás módját, ami lehet impulzus (pulse) vagy tone. Magyarországon ma már gyakorlatilag az összes központ megérti a tone üzemmódú tárcsázást, érdemes ezt választani, mivel megbízhatóbb és gyorsabb.

Ezzel a beállítás végére értünk. Itt az előző lépések bármelyikét megismételhetjük, ha a menüben kiválasztjuk azt. Ha minden rendben, a Finished opció kiválasztásával rögzíthetjük munkánk eredményét. Fentebb említettük, hogy ha a Chat-azonosítást választottunk, felmerülhet még néhány kérdés. Az első ilyen az „Ack String".

Ezt használhatjuk arra, hogy a Connect üzenet után, de még a belépési szkript előtt – esetleg – megjelenő prompton túlléphessünk. Egyes szolgáltatóknál például előfordul, hogy egy host: promptra ppp-vel kell válaszolni, mielőtt a tényeleges belépésre sor kerülhetne. Ebben az esetben a mezőben már látszó két aposztróf mögé írjuk a várt promptot, majd a küldendő válasz szövegét. Szintén a Chat-azonosításnál lesz szükség a „Login



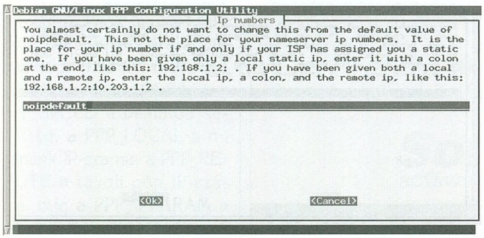
Átjárótan

String" megadására, azaz azon szöveget kell itt megadni, melyben a szolgáltató az azonosítónkat kérdezi. Alapesetben ez „login:", de elég a végét kiírni. Mi „ogin:" beírását javasoljuk, mivel ez a kis és a nagy kezdőbetűs prompttal is működni fog. Ha a szolgáltató „Username:" prompttal kéri az azonosítót, hasonlóan „sername:"-et írjunk be. Ugyanígy történik a „PWD String", azaz a jelszóváró prompt megadása. Az „Other Strings" opciónál általában nem kell semmit megadni, de ha a belépés után további parancsokat kell kiadni, azt itt tehetjük meg. Először jön az a szöveg, melyre várni kell, majd az következik, amit válaszként küldünk.

Kézi beállítás

Nem csak a kényelem kedvéért javasoljuk, hogy minden beállítást a pppconfiggal érdemes kezdeni. Ez egy olyan mintát készít számunkra, ami alapján egyszerűbb elvégezni az egyedi beállításokat. Mint azt az elején már említettük, a szolgáltatóspecifikus beállítások a /etc/ppp/peers, a kommunikációs szkript a /etc/chatscripts alá kerül. Ez utóbbit a chatprogram fogja értelmezni, melyről bővebbet a „man chat" parancs kiadásával kaphatunk.

A /etc/ppp/peers könyvtárban lévő file-ban a pppd szokásos opcióit használhatjuk. A pppconfig által létrehozott file megfelelően taglalt, így a benne található opciókat könnyen megérthetjük. A /etc/chatscript alatti szkriptre a „connect" kezdetű sor hivatkozik. Ez utóbbi szkript formátuma egyszerű chatscript. Vagy opciókat, vagy elvárás–válasz formájú párokat írhatunk bele. Az „ABORT" kulcsszó adja meg azon válaszokat a modemtől, melyekre a kapcsolódást abba kell hagyni, mert hiba történt. Ha egy elvárt vagy egy válasz szöveget nem akarunk kiírni, mert nincs rá szükség, helyette két idézőjelet írjunk. Bármelyik file-t szabadon módosíthatjuk, ezek tárolják a beállítást, és ezt használja a rendszer is.

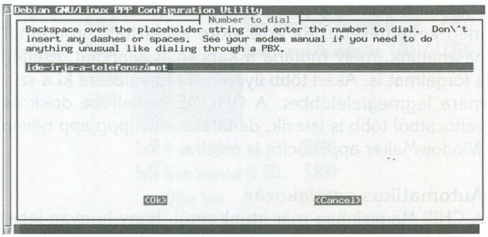


Statikus IP-címet is be lehet állítani

Kapcsolódás

Kézi kapcsolódás

Ha a fenti beállításokat elvégeztük a kapcsolat felépítése egyszerűen a „pon szolgáltató" parancs segítségével történik, ahol a „szolgáltató" a pppconfig programban megadott szolgáltató illetve a /etc/ppp/peers alatti file-név. Ha nem adunk meg a ponnak opciót, akkor az alapértel-



A telefonszám az egyik kulcsparaméter

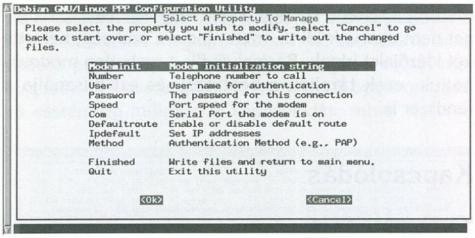
mezett a „provider". A pon parancs igazából egy egyszerű szkript, mely a pppd programot hívja az alábbi módon: „pppd call szolgáltató", ahol a „szolgáltató" azonos a pon parancs paraméterével. Ha kiadtuk ezt a parancsot, a kapcsolat felépítése automatikusan megtörténik, feltéve, hogy jók a beállítások. Ha nem változtattunk az alapbeállításokon, melyet a pppconfig írt be, a chatprogram kimenetét a /var/log/ppp.log file-ban olvashatjuk, illetve ugyanott találjuk a pppd üzeneteit is. Senkit ne té-

vesszen meg azonban az, hogy a pon program azonnal visszatér.

A pppd a háttérben fut, kivéve, ha a „nodetach" opciót megadtuk neki.

A kapcsolatot a „poff szolgáltató" paranccsal szakíthatjuk meg. Itt a „szolgáltató" meg kell egyezzen a kapcsolat felvételekor megadott névvel. Ezen kívül a poffnak vannak még további hasznos opciói, melyekről a „poff -h" parancs által tudakozódhatunk. Az -r opció arra utasítja a pppd-t, hogy szakítsa meg a kapcsolatot, és tárcsázzon újra. A -d opció ki-, illetve bekapcsolja a pppd debug-opcióját, a -c hatására pedig a pppd újraegyezteti a tömörítést.

Ha a grafikus felületen dolgozunk, nem feltétlenül szeretnénk folyton egy terminálablakot indítani a fenti parancsok kiadásához.



A menüből kiválasztva még egyszer megpróbálkozhatunk egy-egy beállítással

A legegyszerűbb megoldás, ha egy menüpontba felvesszük a pon és a poff parancs megfelelő formáját, de még kellemesebb, ha erre előre elkészített programot használunk, mely mutatja a kapcsolat állapotát, esetleg a forgalmat is. Akad több ilyen is, ki-ki válassza ki a számára legmegfelelőbbet. A GNOME paneljébe dokkoló változatból több is létezik, de találunk wmppp.app néven WindowMaker applikációt is e célra.

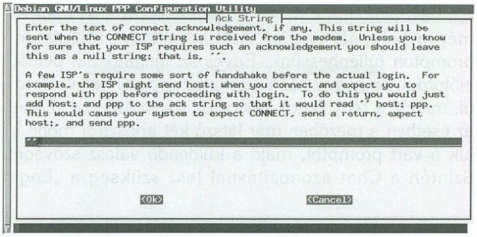
Automatikus csatlakozás

A CHIP Magazinban már írtunk arról, hogy hogyan lehet a diald programot automatikus kapcsolatépítésre használni. Természetesen, ez Debian alatt is megy, a beállításokat a /etc/diald könyvtárban találjuk.

A diald.options beállítása a legfontosabb. Az ugyanott található connect szkript fogja felépíteni a kapcsolatot, de a /etc/chatscript alatti szkriptek is tökéletesek a célra, azaz onnan is vehetjük azt.

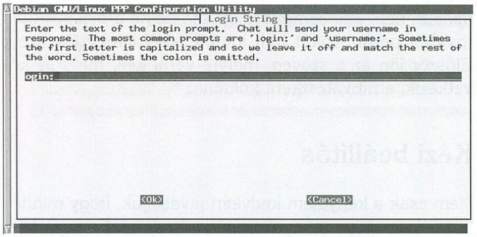
Ennél többet a dialdról nem írunk itt, inkább a 2.2-es Kernellel működő, a pppd szintjén megvalósított módszert ismertetjük.

A legújabb pppd alkalmas a feltételes tárcsázásra,

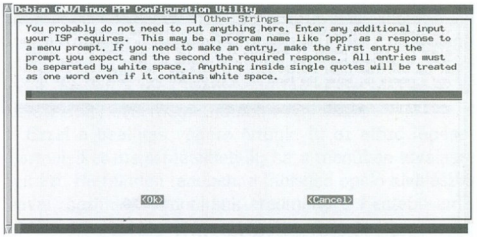


A szerver előzetes promptját léphetjük át az ack string megadásával

azaz arra, hogy ha van IP-forgalom, felépítse a kapcsolatot, ha nincs, lebontsa azt. Ezt alapvetően a demand opció valósítja meg, de szükségünk van a persist és az idle opciókra is. A persist utasítja a pppd-t, hogy folyamatosan maradjon aktív, azaz a kapcsolat bontásakor ne lépjen ki. A demand jelenti azt, hogy szükség esetén tárcsázzon, az „idle idő" pedig azt, hogy ha nincs forgalom, „idő" másodperc eltelte után bontsa a kapcsolatot. Ez alapján, ha a pppconfiggal már beállítottunk egy szolgáltatót, a „pppd persist demand idle 60 call provider" parancs elindítja a pppd-t a kívánt módban, úgy, hogy szükség esetén a „provider" nevű szolgáltatóhoz kapcsolódik, és forgalom hiányában 60 másodperc eltelte után bontja a vonalat. Ennek használatához feltétlenül meg kell adnunk a saját és a távoli gép IP-számát, mivel a



Készül a beléptető szkript



További speciális parancsokat is küldhetünk a szervernek

pppd-nek létre kell hoznia a szükséges PPP hálózati eszközt és az útirányítást (route).

Programok futtatása a kapcsolat felépítésekor és bontásakor

Sokszor szükségünk lehet egy-egy program indítására a PPP-kapcsolat felépülése és bontása során. Például leveleink letöltésére használhatjuk a fetchmail programot. Ez folyamatosan fut, és adott időnként letölti a leveleket, akár több helyről is. Azaz érdemes a kapcsolat felépülésénél elindítani, lebontásánál pedig leállítani.

De ugyanígy tölthetjük le kedvenc hírcsoportjainkat is. Linux alatt a /etc/ppp/ip-up szkript fut le a kapcsolat felépülése után, az ip-down a kapcsolat megszakításakor. Debian alatt ez annyival bővül, hogy a fenti két szkript meghív minden a /etc/ppp/ip-up.d/ illetve ip-down.d/ könyvtárakban található programot. Ennek köszönhetően nagyon egyszerűen adhatunk hozzá vagy törölhetünk szkripteket, illetve programokat.

Néhány program telepítése során ez automatikusan meg is történik. Ilyen például az slrnpull, mely a hírcsoportok letöltését szolgálja. A két szkript (ip-up és ip-down) beállít néhány változót, melyet programjainkban lekérdezhetünk: PPP_IFACE az interfész neve (általában ppp0), a PPP_TTY a soros port vagy más terminál neve (például /dev/ttyS1), a PPP_SPEED a beállított sebesség, a PPP_LOCAL a mi gépünk IP-száma, a PPP_REMOTE a távoli gép IP-száma, míg a PPP_IPPARAM a pppd által átadott további opciók.

Linux alatt a pppd számos finomhangolási opcióval rendelkezik. Lehetséges például IPX-forgalom továbbítása a ppp-kapcsolaton keresztül.

Ezek részletes tárgyalása meghaladja kiadványunk kereteit, de a pppd leírásában megtalálhatók, így mindenkinek csak azt javasolhatjuk, hogy szükség esetén ott tájékozódjon. ■

Slapic
slapic@vogel.hu

Psion és Linux

Napjaink egyik méltán legnépszerűbb tenyérgépe a Psion. Az alábbiakban azoknak szeretnénk néhány tippet adni, akiknek sikerült hozzájutniuk ehhez a nem éppen olcsó aprósághoz. Lássuk, hogyan lehet a készüléket összekapcsolni egy Linuxot futtató számítógéppel.

Alapvetően háromféle kapcsolatról beszélhetünk. Az egyik a TCP/IP, mely leveleink letöltésére, vagy webböngészésre ad lehetőséget. A másik arra szolgál, hogy Linux alól láthassuk a készüléken tárolt adatokat. Emellett egyszerű terminálként is használhatjuk a Psiont, például Linux-szerverek távadminisztrálására. Az alábbiakban elsősorban a Series 5 használatáról írunk, de kis változtatásokkal a leírtak érvényesek a 3-as sorozatra is.

Terminálkapcsolat

Egy Linux-szerver eléréséhez bármilyen vt100 vagy annál újabb terminálemuláció megfelel. Bár a Psion beépített kommunikációs programja, a Comms is alkalmas erre, nem ez a legjobb választás. A flyTERM nevű shareware program sokkal komfortosabbá teszi munkánkat. A Psion oldalán igen egyszerű a kapcsolat konfigurálása: a kívánt sebesség és az adatátviteli formátum (8 adatbit, 1 stopbit, nincs paritás) beállítása után használatra kész a program. A sebességet az átvivőközegtől függően érdemes beállítani. Mobiltelefon használata esetén ezt nem érdemes 19200 fölé vinni, vonalas modem és közvetlen kábelkapcsolat (melyre a készülékhez kapott kábel kiválóan alkalmas) esetén az 57600-at javasoljuk, a 115200 nem mindig működik. Linuxoldalon, ha modemes kapcsolatot szeretnénk létesíteni, ugyanazt a megoldást kell alkalmaznunk, mint minden más modemes kiszolgálás esetén. Közvetlen kábelkapcsolatnál pedig vagy a getty vagy az mgetty program közvetlen módját kell alkalmazni. Debian alatt erre jó példákat találunk a /etc/inittab file-ban, ahova be kell írni a getty indítását végző sort. Íme egy mgetty mintasor a COM1-re csatlakoztatott Psion kiszolgálására:

T1:23:respawn:/sbin/mgetty -r -s 57600 ttyS0

Emellett az alábbi bejegyzéseket kell beírni az mgetty konfigurációs file-jába (/etc/mgetty/mgetty.config) a normál beállításokon kívül:

port ttyS0 direct y speed 57600 toggle-dtr y – .

Ha ezzel végeztünk, máris beléphetünk a Linuxba. Sajnos a sebesség nem éppen szédítő, valamint a soros kommunikáció eléggé megterheli az elemeket, de még így is napokig kibírja egy pár ceruzaelem.

TCP/IP-kapcsolat

Ha modemen keresztül szeretnénk a TCP/IP-kapcsolatot felépíteni, nincs más teendőnk, csak a normál modemes fogadást és PPP-kapcsolatot felépíteni, ahogy azt a PPP- és modemszerverről szóló írásunkban leírtuk.

A Psionról egyszerűen felkapcsolódhatunk a szerverre, és máris működik az Internet-kapcsolat.

Közvetlen kábeles kapcsolatnál két lehetőségünk is van. Ha a fenti módon beállított terminálkapcsolat működik, és ehhez az mgetty-t használjuk, könnyedén felvehetjük a ppp-t is a szolgáltatások közé. Ismét hivatkozva a PPP-szerverről szóló cikkre, az ott leírt automatikus PPP-kapcsolathoz szükséges bejegyzés tökéletesen megfelel a célnak az mgetty login konfigurációs file-jában (/etc/mgetty/login.config). Ha nem fut a soros porton az mgetty, a pppd közvetlen indításával is létrehozhatjuk a kapcsolatot. Így a pppd folyamatosan fut, és minden alkalommal, amikor a Psion kezdeményezi a PPP-kapcsolatot, a vonalon kiosztja az IP-számot, és beállítja a Linuxoldalon szükségeseket.

Érdemes a /etc/ppp/options.ttyS0 (COM1 esetén) file-ba beírni a beállításokat, hogy azokat ne kelljen minden egyes alkalommal újragépelni. Azért nem az általános options file-ba, mert normál modemes kapcsolatokhoz, például a Linux alóli modemes internetezéshez, ezek nem jók. Az alábbi opciókra van szükségünk a közvetlen kábeles kapcsolathoz:

local silent proxyarp crtscts lock persist 192.168.1.1:192.168.1.6.

Az opciók jelentése a fenti sorrendben a következő:

local: közvetlen kábelkapcsolatot használunk; silent: a másik féltől várjuk a kezdeményezést, nem írunk a portra semmit;

proxyarp: a helyi hálózat számára lokálisnak mutatjuk a kiszolgált gépet (így lehet egy alhálóban lévő címet adni a PPP túloldalán levő gépnek);

crtscts: hardware flow control;

lock: a használat idejére kisajátítjuk a portot, más programok nem használhatják;

persist: a kapcsolat megszakításakor a pppd nem lép ki, hanem újra vár a kapcsolat felépülésére;

Kettősponttal elválasztva a saját gépünk IP-számát és a Psionnak kiosztandó IP-számot kell megadni. Ez utóbbi elhagyható, ha a készüléken kézzel állítjuk be.

A PPP-ta /usr/sbin/pppd /dev/ttyS0 paranccsal indítsuk. A Psionon közvetlen kábelkapcsolatot kell beállítani, és a program kérdésére ki kell választani, hogy melyik definiált kapcsolati lehetőséget használja. A name-szervert (DNS) kézzel kell megadnunk. Login szkript vagy autentikáció a fentihez nem kell, de természetesen a Linuxoldalon kérhetünk PAP- vagy CHAP-azonosítást is, ugyanúgy mint más PPP-szervereken.

File-rendszer kapcsolat

Talán a legfontosabb, amire a Psion és a Linux összekapcsolását használni akarhatjuk, az az, hogy file-okat tudjunk mozgatni egyikről a másikra. Jelenleg csak a linuxos gépről láthatjuk a Psionon tárolt file-jainkat, fordítva még nem megoldott a kapcsolat.

Sajnos a Psion beépített remote linkje ehhez nem jó, azt ki kell kapcsolni és egy külön programot kell futtatni, mely nfsc.opl néven forráskóddal együtt elérhető a p3nfs csomagban. Ezt terminálon keresztül egyszerű XModem protokollal át tudjuk tölteni, majd Psion „Program" menüpontjából indítható OPL szerkesztő- és fordítóprogrammal lefordíthatjuk. Ha ezzel végeztünk, és a p3nfs csomagot telepítettük (ez a CD-n megtalálható, a Debian rendszer része), rootként el kell indítatni a linuxos programot. Ha Series 3-hoz kapcsolódunk, semmi különleges paramétert nem kell megadni. A Series 5 használatakor azonban igen, de találunk egy indítószkriptet /usr/lib/p3nfs/etc/p5nfsd néven, mely majdnem tökéletes.

Nekünk 115 200-on nem sikerült életre kelteni a kapcsolatot, ezért a sebességet 57 600-ra csökkentettük. Ehhez az nfsc.opl-ben keressük meg a „KBaud115200" szöveg második előfordulását, ami így néz ki:

rsset:(KBaud115200%, 0, 8, 1, 0, &0).

Ebben cseréljük le a 115200-at 57600-ra, majd fordítsuk újra a programot. Ezután az említett p5nfsd indítóprogramot is javítsuk ki. A setserial /dev/ttyS0 spd_vhi parancsban az spd_vhi helyett spd_hi kell, valamint Debian alatt a p3nfsd elérési útja sem jó, mivel a program a /usr/bin könyvtárban van. Ezután indíthatjuk a Psionon az nfsc.opo-t, majd amint az elindult, a Linuxon rootként a p5nfsd szkripttel a p3nfsd programot.

Ha valaki nem találná a szkriptet, az alábbi parancssort kell begépelnie:

/usr/bin/p3nfsd-tty /dev/ttyS0 -series5 -oldnfsc -wakeup

A „wakeup" egy igen kellemes opció: hatására a készülék bekapcsol, amikor hozzá fordulunk. Ha minden rendben, az alábbi üzenetet kapjuk a Linuxon:

p3nfsd: version 5.3, using /dev/ttyS0 (38400), mounting on /psion/mntp3nfsd: to stop the server do „ls /psion/mnt/exit". (pid 9154)

Senkit se zavarjon a 38 400-as sebességkiírás, mindig ez a szám jelenik meg, de ettől még a kapcsolat a fent megadott sebességen él. A második sor tájékoztat minket, hogy a programból szabályosan, egy egyszerű trükkel, egy sima ls parancscsal léphetünk ki. A kérdéses „exit" nevű file-t senki se keresse, nem létezik, és az se zavarjon meg senkit, hogy „file exist" hibaüzenetet ad az ls. Ettől még a program szabályosan kilép. Ha másképp lépünk ki (például kill), a csatolt file-system látszólag megmarad, mely később sok kellemetlenséget okozhat. A p3nfsd, ugyan semmi köze az NFS-hez (azaz a Network File Systemhez), azt emulálja a Linux felé, és így néz ki a program futásakor a mount parancs kimenetének ide vonatkozó sora:

localhost:/dev/ttyS0 on /psion/mnt type nfs (hard,intr)

Ennek el kell tűnnie, ha a program szabályosan lépett ki. Ha sikerült a kapcsolat felépítése, a /psion/mnt könyvtárban találjuk a Psion drive-okat. Egyszerűen beléphetünk a C: drive-ra, ha bemegyünk a /psion/mnt/C: könyvtárba. Ezután a Linux alatt megszokott file- és könyvtárműveletekkel dolgozhatunk. SIS programtelepítővel ellátott programokat is installálhatunk, ha a .sis file-t felmásoljuk a készülékre és ott elindítjuk.

Adatformátumok

Sok esetben hasznos lehetőség, ha a Psionon futó programokban tárolt adatainkat át tudjuk konvertálni a Linuxon futó programjaink számára. Sajnos, erre még várni kell, de addig is a szövegszerkesztőből sima text formátumban, a táblázatkezelőből vesszővel elválasztott file-formátumban kinyerhetjük az adatokat.

A határidőnapló (Agenda) adataival, sajnos nincs ilyen szerencsénk. A 3-as sorozat formátumát lehet konvertálni, de az 5-ös formátuma ismeretlen. Reméljük, mihamarabb akad egy vállalkozó szellemű egyén, aki az ical-hoz vagy a GNOME Calendarhoz ír egy kiegészítőt, mely olvassa és írja a Psion Series 5 Agenda file-formátumát.

Mit hozhat a jövő?

A fenti formátum konverterek mellett érdemes felfigyelni arra a projektre, mely a Linux portolását végzi a Psion architektúrájára. Egy alap 8 megabyte-os gépen a Linux már igen jól futna. Nem kell azonban megvárni e munka eredményét, már ma is linuxozhatunk a Psionon. Mindössze a Psionhoz beszerezhető PC-emulátorban kell elindítani az i386-os Linuxot. ■

Slapic
slapic@vogel.hu

A 2.2-es Kernel beállítása

Az elmúlt hónapok legnagyobb újdonsága a Linux új, 2.2-es verziójú kernelének megjelenése. Ez az előző stabil (2.0-ás) verzióhoz képest számos újdonságot, plusz funkciót és emellett pár alapvető változást is hozott. Most a teljesség igénye nélkül megpróbálunk egy rövid útmutatót adni, hogyan állítsuk be és telepítsük a kernelt, elsősorban Debian alatt.

Természetesen a változások nem jelentenek komoly kompatibilitási problémákat, de van pár program, melyet mindenképpen frissíteni kell. Erről már írtunk a CHIP Magazinban. A CD-mellékleten megtalálható Debian GNU/Linux 2.1-es disztribúcióban lévő programokat pedig nem kell frissíteni.

Az első lépések

Ha nem vagyunk elégedettek a „gyári" kernellel, vagy – ahogy egy jó linuxostól elvárható – a saját hardverünkre optimalizált kernelt szeretnénk használni, azt újra kell fordítani. Ez nem nagy ördöngösség, Debian alatt pedig különösen egyszerű.

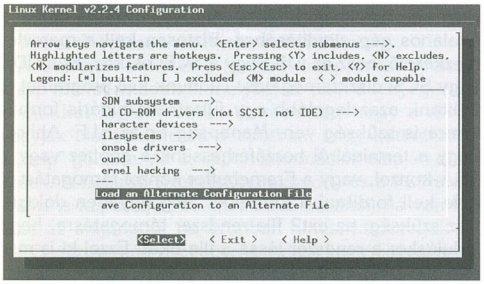
Mindenekelőtt be kell szerezni a szükséges forrásokat, melyek tömörítve(!) 12 megabyte-nál is több helyet foglalnak, a fordításhoz kicsomagolva pedig ennek többszörösét. Fordítás közben akár hatvan megabyte fölé is nőhet a méretük. Erre készüljünk fel, mielőtt nekikezdünk. A forráskód beszerzésére több jó lehetőség is kínálkozik. Interneten az ftp.hu.kernel.org FTP-szerverről tölthetjük le, de megtalálható a CHIP Magazin CD-mellékletein is. Ha Debian Linuxot használunk, nemcsak a bináris kernelt, hanem a forráskódot is feltehetjük a Debian csomagból. Ez a forráscsomag is része a CD-inken megtalálható Debian disztribúciónak.

Ha feltettük a csomagot, a /usr/src könyvtárban találjuk meg kernel-source-2.2.4.tar.gz néven, feltéve ha a 2.2.4-es kernelt tettük fel. Ezt belépve az adott könyvtárba a „tar xzf kernel-source-2.2.4.tar.gz" paranccsal kicsomagolhatjuk. Természetesen máshonnan származó kernelforrást is ugyanígy csomagolhatunk ki. Ha FTP-ről szedtük le a kernelt, minden bizonnyal linux-2.2.4.tar.gz (a 2.2.4-es kernelnél maradva) néven töltöttük le, amit (mivel nem a disztribúció része) a /usr/local/src könyvtárba illik tenni, és a fenti módon kicsomagolni. Azt, hogy miért oda, a CHIP Magazinban megjelent, a 2.0-ás kernel fordításáról szóló írásunkban részletesen is kifejtettük. A /usr/local/src könyvtárat hozzuk létre, ha még nem létezik, de oda akarjuk tenni a forrást.

Most maradjunk a /usr/src alatt és a Debian csomagból feltett kernel forráscsomagnál, hiszen Debian alatt ez a legegyszerűbb. Ha kicsomagoltuk az említett file-ból, a kernelt a /usr/src/kernel-source-2.2.4/ nevű könyvtárban találjuk. Debian alatt szükségünk van még pár csomagra, melyek nagyban megkönnyítik a munkánkat. Természetesen a standard fejlesztői csomagok (gcc, libc6-dev stb.) mellett. Ezek közé tartozik a kernel-package, mely a forrásból bináris Debian csomagot generál, hogy azt a dpkg csomagkezelővel telepíthessük. Működéséhez elengedhetetlen a Perl, a dpkg és a dpkg-dev, valamint a fileutils csomagok. Továbbá javasolt a debianutils telepítése is. A Debian csomagok készítéséhez a devscripts csomag telepítését is ajánljuk.

Konfigurálás

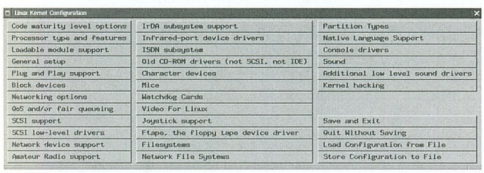
Ha az előkészületekkel végeztünk, be kell állítanunk a kernelt. Jó tudni, hogy az épp futó kernelre a /vmlinuz szimbolikus link mutat, és maga a kernel a /boot könyvtárban található vmlinuz-verzió néven. Ezt a verziót, például 2.2.1, azaz /boot/vmlinuz-2.2.1 néven találjuk meg futó kernelként. Ugyanott (a /boot-ban) van egy config-verzió file is, ahol a verzió azonos az előbb leírtakkal. Ebben a file-ban van az épp futó kernel teljes beállítása, amit akár egy új fordításához is



A Kernel menüs beállítása

használhatunk. Felhívjuk olvasóink figyelmét, hogy a 2.0-ás kernel konfigurációja nem hasznosítható a 2.2-es kernelek beállításához és viszont. Ha használni szeretnénk, a legegyszerűbb, ha a konfigurációs menüben (erről lejjebb írunk) a „Load configuration" opcióval betöltjük azt.

A beállításokat négyféle módszerrel is végezhetjük. Minden esetben lépjünk be a kernel forráskódját tartalmazó könyvtárba, ami a példánkban a /usr/src/kernel-source-2.2.4. Ha kézzel szeretnénk konfigurálni, akkor a .config file-t módosítsuk egy sima szöveges szerkesztőprogrammal, például a vi-jal. Ez a file tartalmazza a beállításokat. Ha még nem létezik, másoljuk ide a már fent említett, a futó rendszer beállításait tartalmazó file-t .config néven, vagy az arch/i386/defconfig file-t (Intel platform esetén). Az ebben a file-ban található beállítási opciókra hivatkozunk azon írásainkban, ahol egy-egy kernelbeállítás részletet mutatunk be példaként egy adott funkció működéséhez.



A Kernel grafikus beállítása

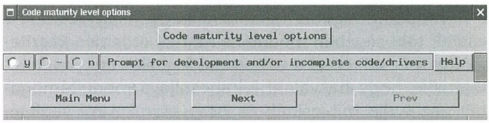
A hagyományos módszer az egyszerű szöveges beállítás, melyet a „make config" paranccsal kezdeményezhetünk. Ehhez hasonló a „make oldconfig" parancs, mely végigfut a már meglévő beállításokon, és csak akkor kérdez, ha az adott kernelnek van olyan funkciója, mely még nem szerepel a .config file-ban, egyébként az ott leírt beállításokat változatlanul hagyja. A már tényleg felhasználóbarát, azaz menüs beállításból kettőt találunk: egy szövegeset és egy grafikust. A szöveges a „make menuconfig" paranccsal a grafikus a „make xconfig" paranccsal indítható.

A két megoldás egyenértékű, és mi most a grafikus változat alapján haladunk végig a fontosabb opciókon.

Általános beállítások

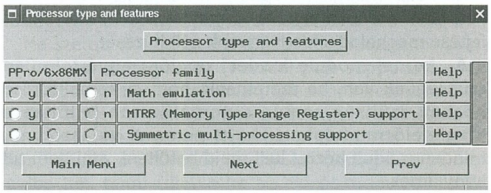
Ide sorolhatjuk a „kódkiforrottsági-szint" (Code maturity level options) beállítását, a processzor (Processor type and features), a modul támogatás (Loadable module support) és az általános beállítások részt (General setup), valamint a Plug and Play támogatást (Plug and Play support).

A kódkiforrottság-beállítás magyarul annyit jelent, hogy kívánjuk-e használni a még nem teljesen stabilnak („EXPERIMENTAL") tekintett szolgáltatásokat.



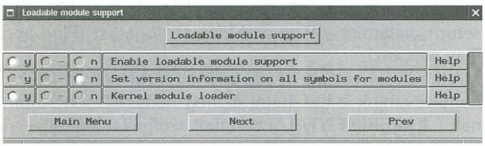
Mi csak akkor javasoljuk itt az igen („y") választ, ha feltétlenül szükség van egy ilyen alkalmazásra.

A processzor beállításnál adhatjuk meg, melyik processzorra legyen optimalizálva a lefordított kernel.



Itt a 386-ostól a Pentium II-ig teljes a választék. A Pentium II/III, PPro, Xeon és Celeron processzorokhoz a PPro opciót válasszuk, de ide tartozik a 6x86MX és MII processzorcsaládok minden tagja is. Az AMD K5/K6, Intel Pentium és Pentium MMX processzorok a „Pentium" csoportba tartoznak, míg a régebbiek értelemszerűen a 486 vagy 386-ba. A „Math emulation" opcióra azoknál a régi processzoroknál van szükség, melyek még nem tartalmazták a matematikai koprocesszor funkcióit, és külön sem rendelkezünk ilyennel. Minden egyéb esetben kapcsoljuk ki a funkciót. Az MTRR PCI vagy AGP buszon kommunikáló eszközök speciális memóriakezelését segíti. Ez akár két és félszeres sebességnövekedést hozhat grafikai megjelení-

téskor, ha az X-szerver ezt ki is használja. Érdemes bekapcsolni. A „Symmetric multi-processing support" a többprocesszoros gépek esetén szükséges annak érdekében, hogy a kernel használja az összeset. Ha nem kapcsoljuk be, futni fog ugyan a kernel többprocesszoros gépeken, de csak egyetlen processzort fog használni. Ha bekapcsoljuk, és egyprocesszoros gépen használjuk az így elkészített kernelt, vagy fog futni, vagy nem, és biztosan lassabb lesz, mint ha SMP nélkül fordított kernelt használnánk.



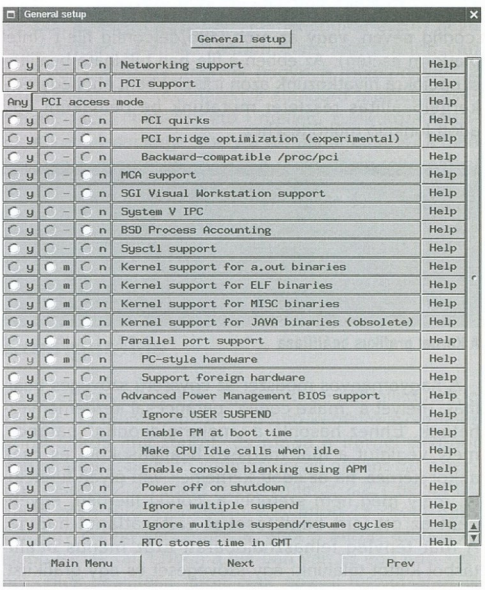
A modultámogatás beállítása nagyon egyszerű. Az esetek többségében moduláris kernelt érdemes készíteni, így mi csak ezzel a változattal foglalkozunk, és feltételezzük, hogy ha valaki kikapcsolja a modultámogatást, az tudja mit csinál, és nem szorul a mi segítségünkre. Azaz az „Enable loadable module support" kérdésre a válasz „y". A verzió információk beállítása a modulokban (Set version information on all symbols for modules) nem ajánlott. Ha egyszerre akarunk azonos verziószámú kerneleket többféle modullal használni, vagy külső modulokat építünk a kernelhez hasznos lehet. Ezek azonban olyan szakmai kérdések, melyek kifejtése meghaladja kiadványunk célkitűzését.

A „kernel module loader" opcióra egyértelműen szükségünk van, ha szeretnénk, hogy a kernel futása közben szükség esetén automatikusan betöltse a megfelelő modulokat. Ha itt nemmel válaszolunk, minden modult kézzel kell majd betölteni, ami nem túl kényelmes.

Itt érdemes szót ejteni a modulok használatáról. A Linux tudásának igen nagy részét fordíthatjuk modulként, és így azok nem növelik feleslegesen a futó kernelt. Ezen modulok egy részére tekinthetünk úgy, mint máshol az eszközökhöz adott driverprogramokra, de ezek annál sokkal összetettebbek. Az tény, hogy a kernel által támogatott hardveregységeket is ezek kezelik. Egy kernel építésekor egy alapelvet mindenképpen érdemes követni: csak azt fordítsuk bele a kernelbe (y opció), amire a rendszer indításakor szükségünk van, minden mást tegyünk modulba.

Érdemes olyasmiket is modulba rakni, amikre várhatóan nem lesz szükségünk, de ha mégis, nem akarunk emiatt kernelt fordítani. Szükség esetén csak beállítjuk és betöltjük a modult (ha a kernel nem teszi meg automatikusan). Gondoljuk végig, mi is kell egy általános gép elindításához. Biztosan kell a merevlemezkezelő, azaz az IDE vagy a SCSI eszköz, és az IDE vagy SCSI diszktámogatás. Tudnunk kell programot is indítani, azaz legalább egy futtatható bináris formátumra is szükség van. Manapság ez az ELF. Ahhoz, hogy a terminálról hozzáférhessünk a géphez vagy a VGA-konzol, vagy a Framebuffer konzol támogatást is bele kell fordítani a kernelbe. Még egyetlen dologra lesz szükség: az ext2 file-rendszer támogatásra, hogy induláskor a rendszer lássa a file-okat. Ezzel ki is merült egy átlagrendszer alapvető igénye. Természetesen sok olyan funkció van, melyet nem lehet modulba tenni, és be kell kapcsolni. Ezekről majd az adott opcióknál írunk.

Érdemes felfigyelni rá, hogy nem volt szükség sem CD-ROM-, sem floppy-támogatásra, és ez nem tévedés. Még ha floppyról indítjuk is a rendszert, nem kell, hogy benne legyen a kernelben annak támogatása, hiszen azt még a LILO tölti be és indítja el a BIOS segítségével. Egy érdekesség az úgynevezett diskless, azaz merevlemez nélküli munkaállomások indítása. Egy ehhez megfelelő kernelbe nem kell a Linux natív ext2 file-rendszer támogatása, de kell az NFS, a hálózati kártya és a hálózati automatikus konfiguráció, hogy a hálózatról csatolni tudja a file-rendszert. Szintén felesleges



ilyenkor akár SCSI, akár IDE támogatást tenni a kernelbe: ezek teljesen jók lesznek modulként.

Az általános beállítások (general setup) már egy hosszabb menüt mutatnak, melyből itt csak a lényegesebb opciókkal foglalkozunk. Javasoljuk, hogy a többit hagyják alapértelmezett állapotban, vagy kattintsanak a „help" gombra. Az itt kapott információk alapján döntsenek. A hálózati (Networking support) és a PCI támogatást is kapcsoljuk be. Abban az esetben, ha van PCI busz az alaplapon, ha azonban nincs, kapcsoljuk ki. A PCI elérési mód (PCI access mode) segítségével megadhatjuk, hogy a Linux hogyan keresi a PCI eszközöket. Több választási lehetőségünk is van. Kizárólag a BIOS-on keresztül történik az elérés. Ez egyes hibás BIOS-szal rendelkező gépek esetén lefagyáshoz vezethet. Bár vannak PCI BIOS nélküli gépek is, ahol egyáltalán nem működik. A Direct opciót választva a kernel a BIOS-t megkerülve közvetlenül próbál szóba állni a PCI eszközökkel, míg az „Any", azaz bármely opciót választva először közvetlenül, majd ha az nem sikerül a BIOS-on keresztül próbálkozik. Mi ez utóbbit javasoljuk.

Az egyik jelentős változás az új kernelben a PCI eszközkezelés és annak elérhetősége programból. Az új kernelben már a /proc/bus/pci-ban vannak a PCI információk, de számos program igényli még a régit, az a /proc/pci létezését. A „Backward-compatible /proc/pci" opcióval pont ezt kapcsolhatjuk be, és ilyenkor egyszerre meglesz mindkét eszköz.

Az MCA támogatás a mikrocsatornás (például PS/2) gépekhez szükséges, ennek hiányában kapcsoljuk ki. Hasonlóan speciális gépekhez, a Silicon munkaállomásokhoz használjuk az „SGI Visual Workstation support" opciót. Programok egymás közötti kommunikációjára használatos az IPC, azaz az Inter Process Communication, és erre nagy szükségünk lesz, ezért kapcsoljuk be. A BSD process accounting opció már sokaknak okozott fejfájást. Főleg fordítás közbeni hibákkal kecsegtet néhány 2.1-es kernelben, de a példaként használt 2.2.4-ben is. A 2.2.5-ben ezt a hibát már kijavították, így ott, ha szükségünk van a processzek fogyasztásának számlázására, bekapcsolhatjuk. A Sysctl támogatást is javasoljuk, hogy kapcsolják be. Ennek haszna, hogy futás közben lehet bizonyos paramétereit változtatni a kernelnek (/proc/sys).

A beállítások egy nagyon lényeges pontja a futtatható bináris támogatás. Egy speciális adottsága a Linuxnak, hogy eleve több bináris formátumot tud futtatni natív módon, de kernel támogatással és megfelelő segédprogrammal (emulátorral) bármit. Ennek támogatását állíthatjuk itt be. Az „a.out" a régi, libc4-gyel használt bináris formátum. Csak akkor van szükség a bekapcsolására, ha van olyan ősködvület programunk, amit nem tudunk újrafordítani az új libc5 vagy libc6 rendszerekhez, és mégis használni szeretnénk. Nyugodtan tegyük modulba („m"), így ha szükség van rá betölti a kernel, egyébként pedig nem foglal memóriát.

Amint azt a moduloknál már írtuk, az ELF támogatás elengedhetetlen, hogy a rendszerünk elinduljon. Persze indíthatjuk a rendszert a.out binárisokkal is, de kinek van erre ma már szüksége? Egy nagy ötlet a „MISC" bináris támogatás, mely teljesen általános a kernelhez. Segítségével akár Java, DOS vagy Windows programokat is indíthatunk parancssorból, ha beállítottuk a dosemu vagy a wine indítását. Ha ezt modulba tesszük, a modult betöltjük és a /proc/sys/fs/binfmt_misc/register file-ba beírjuk a megfelelő sort, máris megy. Például windowsos programok kényelmes indítását wine-nel az alábbi módszerrel tehetjük lehetővé:

```
echo ':DOSWin:M::MZ::/usr/bin/X11/wine:' >
/proc/sys/fs/binfmt_misc/register
```

Az echo parancs kiírja a mögé írt szöveget, melyet a „>" jellel beleirányítunk az említett file-ba, mely jól láthatóan a /proc/sys alatti, azaz a már említett futás közbeni vezérlőket tartalmazó eszköz. Az echo paramétereként szereplő szöveg határozza meg, hogy milyen bináris milyen segédprogrammal futtatható. A példában a wine-t hívjuk meg a Windows programokra, melyeket az „MZ" string azonosít. Erről bővebben a Documentation/binfmt_misc.txt file-ban lehet olvasni.

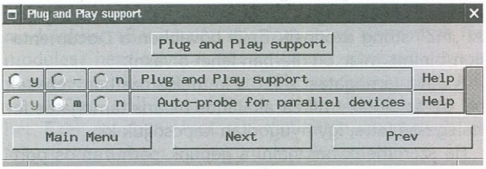
A Java támogatást a 2.0.x kernelsorozatban vezették be, de az előbb említett általános bináris támogatás feleslegessé tette, így nyugodtan kapcsoljuk ki.

Ha szeretnénk használni a gépünk párhuzamos portját, annak támogatását (Parallel port support) tegyük modulba. A már leírtak miatt felesleges betenni a kernelbe, hiszen a párhuzamos portot nem használjuk állandóan, feleslegesen foglalná a memóriát. Ha PC-t használunk, a „PC-style hardware" támogatást is modulba kell tenni (illetve ha az előzőt modulba tettük, csak oda tehetjük, vagy kikapcsolhatjuk). Az idegen hardver támogatást (Support foreign hardware) csak akkor kapcsoljuk be, ha speciális párhuzamos portot akarunk használni (az ECP, illetve EPP portok a szabványos kategóriába sorolandók). Sebességcsökkenést jelent a használata, így ha nem kell, kapcsoljuk ki.

Eljutottunk az APM-hez (Advanced Power Management), azaz a tápkezeléshez. Ez egy igen érzékeny pontja a mai PC-knek, mely sok galibát és váratlan lefagyást tud produkálni. Az alábbi opciókból csak azokat kapcsoljuk be, melyekre feltétlen szükség van, és alaposan teszteljük le a működésüket. A PC BIOS-ában az általános APM támogatáson kívül semmi más tápkezelő funkciót ne engedélyezzünk. Az általunk legtöbb esetben stabilnak talált opciók az „Enable PM at boot

time", mely bekapcsolja az APM támogatást amint a kernel elindult, a konzol (monitor) kikapcsolása az APM segítségével (Enable console blanking using APM) és a gép kikapcsolása a rendszer leállításakor (Power off on shutdown). A többi opciót vagy nem teszteltük, vagy gyakori lefagyást eredményezett egyes rendszereken. Ezekkel óvatosan kísérletezzünk! Ha sikerült mindent jól beállítani, akkor garantáltan stabil rendszerünk lesz. Ami itt még fontos, az az óra kezelése. Telepítéskor a rendszer már rákérdezett, hogy milyen időt tárol a gépünk hardver órája (RTC), és ott írtuk, hogy javasolt a GMT, mivel az mindig egy fix viszonyítási alap, és a helyi időt majd a rendszerünk kiszámolja az időzóna alapján. Itt a kernel szintjén állíthatjuk be ugyanezt, ami arra jó, hogy leállításkor visszaírja a rendszerórát, melyet futás közben például internetes time-szerverekhez igazíthatunk (xntp3 csomag). Tehát az időt lehetőleg GMT-ben adjuk meg a BIOS-ban, és itt kapcsoljuk be az „RTC stores time in GMT" opciót.

Az általános beállítások utolsó lépése a magától működő (Plug and Play) eszközök beállítása a kernel szintjén.

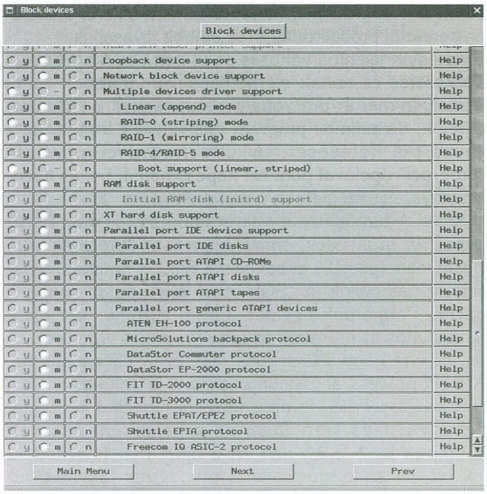


Jelenleg itt csak a párhuzamos port támogatását találjuk, amit érdemes bekapcsolni, valamint az automatikus keresést (Auto-probe for paralell devices), amit tegyünk modulba. Az, hogy itt csak ezzel találkozunk, nem jelenti azt, hogy a Linux nem kezeli a PnP eszközöket, csak nincs rá szükség, hogy ezen bármit beállítsunk, hiszen azért PnP. A BIOS szintjén fontos, hogy a PnP beállításoknál a BIOS hatáskörébe utaljuk az erőforrások (címek, megszakítás) kiosztását (Award BIOS-nál a „PnP OS Installed" opciót „No"-ra állítsuk), így a Linuxnak csak le kell kérdeznie a PCI eszközöktől, hogy milyen címen és megszakításon kezelje őket.

A régi (mondhatnánk nem túl működőképes) ISA-s PnP eszközöket nem kernelszinten kell kezelni, azokhoz az isapnptools csomag programjait használjuk. Ilyen kártyák esetén (például ilyenek az ISA-s ne2000 kompatibilis kártyák, és jópár hangkártya) annak driverét modulba kell tenni, és a modul betöltése előtt az isapnp programmal be kell állítani. Ehhez a /etc/isapnp.conf file-t használhatjuk, melyet a pnpdump (Vigyázat, ez a gép lefagyását okozhatja!) paranccsal készíthetünk el: „pnpdump > /etc/isapnp.conf". Ennek beállítása elég egyedi kártyánként. A pnpdump beleírja a file-ba a kártya által visszaadott valamennyi lehetséges értéket, természetesen „kikommentezve", azaz ezek nem érvényesülnek, de jó alapok ahhoz, hogy kézzel beállítsuk azt, amit szeretnénk. Ha beállítottuk, az isapnp program (mely Debian alatt automatikusan lefut a rendszerinduláskor, ha van /etc/isapnp.conf) beállít mindent.

„Block" eszközök

Ebbe a kategóriába tartozik minden olyan eszköz, melyet blokk módban kezelünk, azaz elsősorban a háttértárak, vagy háttértárat emuláló funkciók; a RAID és a külső eszközök. Kivételt képez a SCSI támogatás, mely külön menüben kapott helyet.



A Normal PC floppy disk supportot, amint azt a korábban írtuk, tegyük modulba, hiszen a rendszer indításához nem szükséges, de később várhatóan kell majd a hajlékonylemezes egység. A nagykapacitású lemezegységek (Iomega ZIP IDE változata, illetve LS-120 avagy A:Drive) kezelését nem ez a modul végzi, hanem az IDE/ATAPI Floppy support opciónál szintén modulba teendő. Az IDE támogatásra is szinte minden rendszeren szükségünk lehet.

Ha IDE eszközről vesszük a rendszer root file-rendszerét, az IDE támogatást a kernelbe kell fordítani, ha

nem (NFS-ről vagy SCSI lemezről), akkor elég modulba. Természetesen ki is kapcsolhatjuk, ha biztosan nincs és nem is lesz IDE eszköz a gépben. Ha használjuk, két választásunk van: egy fejlett (Enhanced) és egy régi (Old harddisk) kezelő. A régit csak merevlemezekhez használhatjuk, és csak abban az igen ritka esetben lehet rá szükségünk, ha egy olyan régi gépen használunk Linuxot, melyen az új driver nem működik, vagy annyira spórolni kell a memóriával, hogy a fejlettebb driver már túl nagy (13 kilobyte-ot takaríthatunk meg). Egyszóval ha nincs rá különösebb okunk, válaszszuk az „Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support" opciót. Mivel ez még csak az általános támogatás volt, be kell állítanunk az egyedi támogatást a különböző IDE egységekhez, mint a merevlemez, cdrom, szalag és floppy. Mint említettük, IDE diszkről induló rendszernél annak benne kell lennie a kernelben, míg az összes többit szerintünk modulba érdemes tenni. Ezek kezelését a kernel teljesen automatikusan meg fogja oldani modulból is.

Lehetőségünk van néhány speciális IDE chipsetet támogató, illetve hibás chipsetek kezeléséhez alkalmas opció kiválasztására is. Ezeket csak akkor kapcsoljuk be, ha az opcióban megnevezett chipsetes alaplapunk van. A mai új alaplapok többsége Intel Triton chipsettel készül, így ezen opciókra nem lesz szükség. A „Generic PCI IDE chipset support"-ra akkor van szükségünk, ha a rendszerben van IDE meghajtó. Ez teszi lehetővé, hogy a kernel automatikusan konfigurálja az(oka)t. Ha csak SCSI egységek vannak, kapcsoljuk ki. Ha nem vagyunk biztosak a dolgunkban, a legjobb bekapcsolva hagyni. Alatta találunk további opciókat, melyek csak ennek bekapcsolása esetén élnek. Ha az alaplapunk tudja a bus-master DMA-t, azt itt kapcsoljuk be. A legtöbb PCI alaplap tudja ezt. A „boot off-board chipsets first" opció lehetővé teszi, hogy olyan IDE eszközről indítsuk a rendszert, mely nem az alaplapi vezérlőhöz, hanem egy kártyához kapcsolódik. Alapesetben az alaplapi vezérlő kapja az ide0 és ide1 eszközneveket, míg egy esetleges külső eszköz az ide2-őt és ide3-at. Ha ezt bekapcsoljuk, akkor a sorrendet cseréljük fel. Fontos, hogy ez átalakítja a merevlemezek kiosztását (hd* eszközök), hiszen a külső kártyára kötöttek kerülnek előre (hda/hdb/ stb.). Ha nem akarunk külső IDE kártyáról bootolni, nem kell bekapcsolni.

Az előző kernelekben a DMA használata automatikus volt, de ez okozott problémákat. Itt most lehetőségünk van eldönteni, hogy a DMA automatikusan bekapcsoljon-e, vagy kézzel kelljen bekapcsolni egységenként. Természetesen, ha automatikusan be van kapcsolva, kézzel ki lehet kapcsolni. Például VIA VP2 chipseteknél igen gyakoriak a DMA-val kapcsolatos problémák, ha ilyen chipset van a gépben, kapcsoljuk ki. A legtöbb esetben nyugodtan bekapcsolhatjuk, és csak akkor érdemes kikapcsolni, ha hibákat észlelünk a DMA körül. A további IDE opciók az egyedi chipsetek még fejlesztés alatt álló driverei. Nem érdemes használni őket, ha nincs rájuk feltétlenül szükség.

Egészen a „loopback device support" opcióig a fenti IDE lehetőségeket találjuk. A loopback egy olyan speciális eszköz, mely egy file-t képes blokk eszközként mutatni, azaz csatolhatunk a rendszerhez floppy- vagy merevlemez image-et tartalmazó file-t. Ez miért jó? Például miután az mkisofs-sel vagy mkhybriddel generáltunk egy CD-image file-t, annak CD-re írása előtt ellenőrizhetjük tartalmát. De készíthetünk floppyra vagy merevlemezre írható file-t is anélkül, hogy lenne ilyen eszközünk, vagy a floppy lassúsága korlátozna minket. Ha kész, a dd paranccsal simán felírhatjuk a file-t a lemezre és kész. Egyszóval érdemes modulba tenni.

A network block device egy igen speciális eszköz. Megfelelő programmal, a hálózati file-rendszerekhez hasonlóan blokk eszközöket, azaz floppykat, CD-ROM-ot vagy merevlemezt is felajánlhatunk a hálózatnak. Az így közzétett eszköz használatára való ez a funkció. Tegyük modulba, ki tudja, hátha szükség lesz egyszer rá.

Amellett, hogy olcsó, a Linuxszal építhetünk komoly és megbízható szervereket anélkül, hogy hatalmas összegeket költenénk RAID eszközök vásárlására, hiszen a Linux megcsinálja nekünk ezt szoftverből. A „Multiple devices driver support" opciót bekapcsolva előtűnnek az egyes RAID változatok driverei. A legegyszerűbb soros (linear) eszközkezeléstől, mely folyamatosan tölti meg az összekötött diszkeket, a nagy biztonságot adó RAID-1-en keresztül a RAID-5-ig több variáció áll rendelkezésünkre. Ha több egyforma diszket szeretnénk egyben, egy file-rendszerként látni, és a linearral szemben komoly sebességnövekedést akarunk elérni, akkor a válasz a RAID-0, azaz a striping mode.

Ha nagy biztonságban akarjuk tudni adatainkat, azaz két egyforma diszket akarunk használni, egyszerre tárolva mindkettőn pontosan ugyanazokat az adatokat, akkor a RAID-1, azaz tükrözés (mirroring) a megoldás. Köztes lehetőség, a RAID-4 vagy RAID-5. Mindkettő több diszken elosztva tárolja az adatokat biztonsági információkkal együtt, ami lemezhiba esetén segít visszaállítani a hibás területen lévő adatokat. A RAID-4 egy külön diszket használ erre, míg a RAID-5 a biztonsági információkat is elosztva tárolja, így hatékonyabbá téve a rendszert. Lehetőségünk van egyes RAID csoportokról bootolni is. Ha bekapcsoljuk a „Boot support" opciót, linear vagy striped (RAID-0) file-rendszerről is indíthatjuk a rendszert. Természetesen ehhez a megfelelő RAID támogatást a kernelbe kell fordítani, míg

egyébként elegendő modulba.Egy másik lehetőség RAID eszközről történő bootolásra az initrd használata, de erről majd később.Mivel partíciókat köthetünk RAID-be, egy diszken lehet normál és RAID-es filerendszer is. Azaz a root filesystem kerülhet normál helyre, míg minden más RAID-esre. Akár többféle RAID-et is alkalmazhatunk egyszerre, de módunk van RAID eszközt újabb RAID-be fűzni, ha épp erre vágyunk.

Az előbb már említettük az initrd nevet. Ideje megmagyarázni, mi is ez. A Linuxban lehet memórialemezt (RAM-diszk) használni. Ez egy egyszerű emulált blokk eszköz, melynek adatai a memóriában vannak. Erre is kedvünk szerint készíthetünk file-rendszert. Ha ezt a kernelbe tesszük, bekapcsoljuk az initrd, azaz „Initial RAM disk support" opciót, lehetőségünk nyílik arra, hogy a kernel indulásakor egy file-t töltsünk be a memória egy meghatározott részére, és azt használjuk root file-rendszerként. Ezt hívjuk initrd-nek. Ha ez elsőre nem is látszik, ennek számtalan haszna van. Ha csak visszamegyünk a fenti RAID lehetőségekhez, máris beláthatjuk. A kernelbe még kevesebb dolgot kell befordítani, még hordozhatóbb, modulárisabb lesz, és a hardvertől szinte teljesen függetlenül indítható lesz a Linux. Az így elindult rendszer azután erről a RAM-diszkről betölti a szükséges modulokat, ami lehet egy RAID vagy egy SCSI, esetleg IDE driver, majd miután így már látja a tényleges root filesystemet tartalmazó eszközt, a RAM-diszk helyett azt csatolja eldobva a memóriából a RAM-diszket. Jól látható, hogy ezzel bármilyen RAID eszközről indíthatjuk a rendszert, vagy anélkül, hogy kernelbe lenne fordítva a drivere detektálhatjuk az épp a gépben lévő SCSI kártyát, amelyhez az a diszk is kapcsolódik, melyen a rendszerünk van. Ilyen initrd-t használ a legtöbb Linux disztribúció telepítője is saját maga elindításához. Tehetjük modulba is, de ilyenkor a bootoláshoz nem jó.

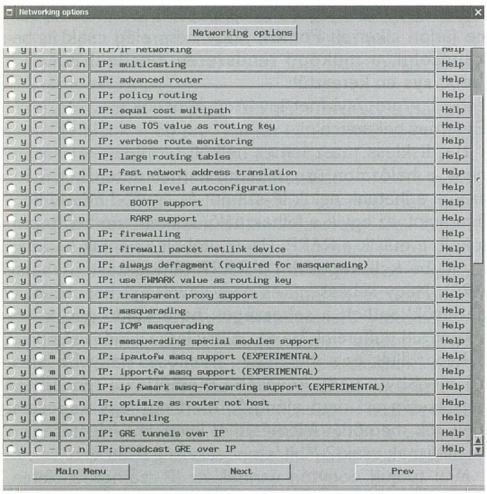
Ha sikerült a múzeumból XT merevlemezt szereznünk, akkor kipróbálhatjuk, milyen másodperceket várni pár kilobyte betöltésére az „XT harddisk support" segítségével. Nyugodtan modulba tehetjük, ha kell megvan, ha nem, legalább nem foglal memóriát (csak pár kilobyte-ot a diszkünkön). Ha valaki ilyen diszkről szeretné indítani a gépét (ráérős típus...) az tegye be a kernelbe.

Még egy újdonság a 2.2-es kernelben: a párhuzamos portra csatlakoztatható IDE eszközök – egy újabb kísérlet az ipar részéről, hogy mobil adathordozót készítsenek a meglévő eszközökkel. Nem teljesen reménytelen, de volt már jobb ötletük is. Manapság az USB az aktuális divat e téren. Addig is, míg lesz USB támogatás a Linuxban – és egyáltalán lesznek működőképes USB eszközök – maradjunk a párhuzamos portra köthetőknél. Mivel ez egy elfogadható kommunikációs sebességre alkalmas csatlakozás (legalábbis az ECP+EPP eszközök), használható a célra. A CHIP Magazinban már említettük az Iomega ZIP drive párhuzamos portos változatát és annak használatát (erről majd a SCSI résznél). Itt az ehhez hasonló, csak IDE protokollt alkalmazó eszközökkel barátkoztathatjuk meg Linuxunkat. Ezek lehetnek diszkek, CD-ROM-ok vagy egyéb IDE eszközök. Természetesen itt sem (ahogy a PC világában már sajnos megszoktuk) voltak képesek a hardvergyártók egy egységes szabvány kialakítására (hm... kész csoda, hogy a SCSI elterjedt a PC-k világában is: túl szabványos), azaz minden gyártónak megvan a maga protokollja az eszközével. Jó néhányhoz megtaláljuk itt a drivert, amit nyugodtan tegyünk modulba. Akár az összeset is modulba tehetjük, az a pár kilobyte a merevlemezen nem számít, és ha szomszédunk, ismerősünk vagy üzletfelünk egy ilyennel állít be hozzánk, újraindítás nélkül azonnal használhatjuk az eszközét, ha az utaztatást túlélte a lemez ;).

Hálózati opciók

Mi más lenne az elsődleges feladata egy Linuxnak, mint a hálózattal kommunikálni? Egyáltalán, milyen ma egy rendszer ami ezt nem tudja? A Linux tudja. Nem csak tudja, olyan szinten tudja, mely tudással kevesen dicsekedhetnek el a világon. Lehet, hogy nagyképűen hangzik, de a tények ezt bizonyítják. Nemcsak egy sima szervert építhetünk Linuxszal, mely több ezer klienst szolgál ki, vagy webszervert, mely óránként százezres találatot kap, hanem tűzfalat vagy útválasztót (router) is, mely beszéli a routerek közös nyelvét, azaz jól beilleszthető egy meglévő router-parkba. Mindezt a policy-routing opcióval megspékelve igen profi rendszert kapunk. Lássuk mit kell ehhez a kernelben beállítani.

Ha belépünk a „Networking options" menübe, a fentebb leírtakat igazolandó egy igen hoszszú lista jelenik meg. Az első opció (Packet socket) azon alapszintű hálózaton kommunikáló programok számára hasznos, melyek nem egy magasabb hálózati protokollt beszélnek. Ilyen például a tcpdump, amellyel a forgalmat figyelhetjük. Érdemes modulba tenni, mert nincs rá állandóan szükség, viszont nagyon hasznos funkció. A „Kernel/User netlink socket" opcióra szinte biztosan szükségünk van, azaz kapcsoljuk be. Ezt használják a programok bizonyos belső kommunikációra a kernel hálózatkezelő részével. Ez segít hozzájuttatni minket útválasztási információkhoz, a tűzfal támadás jelzéseihez, de ezt használja az arpd is, mely segít csökkenteni a feleslegesen nagyra növő ARP táblákat (ezen táblák tar-



talmazzák, hogy mely hardver címhez, mely IP-szám tartozik).

Ahogy az előbb említettük, lehetőség van az útválasztási üzenetek figyelemmel kísérésére. Erre szolgál a „Routing messages" opció. Ha bekapcsoljuk, hozzunk létre egy /dev/route karakteres eszközfile-t 36-os major, 0-ás minor számokkal (mknod /dev/route c 36 0), amiből egy programmal kiolvashatunk jó néhány hálózati útválasztással kapcsolatos információt. Akkor kapcsoljuk be, ha szükségünk van erre a funkcióra.

A „Netlink device emulation" opció csak a visszamenőleges kompatibilitást szolgálja. Egyelőre kapcsoljuk be. Később ez az opció meg fog szűnni. Ha tűzfalat építünk, vagy csak szükségünk van pár tűzfal funkcióra, például a saját védelmünk miatt, kapcsoljuk be a „Network firewalls" opciót. Erre van szükségünk az IP-maszkoláshoz is, hogy csak egy példát említsünk. A „Socket Filtering" funkciók a BSD hasonló funkcionalitásából kerültek ide. Lehetőségünk van a TCP kivételével minden socket kommunikáció szűrésére (TCP/IP csomagszűrést máshol találunk). Ha tudjuk, hogy kell, kapcsoljuk be. Ellenkező esetben tekintsünk el ettől. A Unix programok hagyományosan a unix socketet használják az egymás közötti kommunikációra egy gépen belül (unix domain sockets). Ez egy speciális file, melyet a programok maguknak hoznak létre, és itt várják a kliensek kapcsolódását. A TCP/IP mellett ez a legelterjedtebb kommunikációs módszer Linuxban, így kapcsoljuk be, vagy tegyük modulba (tökéletes modulként, bár a rendszer indításakor szinte azonnal betöltődik és ott is marad). Ezt használja (a TCP/IP mellett) például a Postgresql adatbázis-kezelő is, hogy a helyi kliensekkel szót értsen. A következő a „TCP/IP networking". Azt hiszem nem kell különösebb magyarázat ahhoz, hogy miért kell ezt befordítani a kernelbe.

Ha a TCP/IP-t bekapcsoltuk (és ugye bekapcsoltuk?!), számos IP funkciót társíthatunk hozzá. Ezekkel módjával kell bánni, mert csak a kernelbe fordíthatjuk őket, azaz ha bekapcsoltuk ezeket, akkor foglalni fogják a memóriát, miközben egy átlag rendszeren kevés hasznát vesszük képességeiknek. Ezek a funkciók valósítják meg az útválasztást, a tűzfal funkciókat, de itt állíthatjuk be azt is, hogy a kernel indulásakor a hálózattól bootp vagy rarp protokollal kérje le a saját hálózati címét és a hálózati beállításokat (kernel level autoconfiguration). Ha gépünket egy MBONE hálóhoz kívánjuk kapcsolni, szükség lesz a multicasting funkcióra, mely egyszerre, egy csomaggal több gép megcímzését teszi számunkra lehetővé.

Az „advanced router" opció bekapcsolása lehetővé teszi, hogy további, útválasztással kapcsolatos opciók közül válasszunk. Ezek: „policy routing", azaz a hagyományos útválasztással szemben, ahol csak a címzettől függ, hogy merre tovább, lehetőségünk lesz a döntésbe belevonni a csomag forráscímét is, mint információt. Ha ehhez bekapcsoljuk a „use TOS value as routing key" opciót is, a csomag TOS (Type-Of-Service) információját is felhasználhatjuk. Egyszóval igen kellemesen irányíthatunk sok hálózatot magunkon keresztül. A „verbose route monitoring", egy olyan ajánlott funkció, mely számos információt ír a rendszer naplófile-ba (klogd-n keresztül). Ez segít észrevenni egy a hálózaton rosszul konfigurált rendszert. Ha nagy zónák számára készítünk útválasztót, kapcsoljuk be a „large routing tables" opciót, mely az alap 64-nél több útválasztási bejegyzés tárolását is lehetővé teszi. Sokszor volt már szó a NAT-ról, azaz a Network Address Translationról. Ha bekapcsoljuk a „fast network address translation" opciót, képessé tesszük a kernelt az átmenő csomagok forrás- és célcímeinek módosítására.

Tűzön-vízen át

Egy izgalmas téma következik: az IP tűzfalak. Ezt használhatjuk csomagszűrő tűzfalak építésére, de ez kell az IP-maszkoláshoz is. A CHIP Magazinban többször szerepelt ipchains program alkalmas ezen funkciók kezelésére. Ha bekapcsoljuk, lehetőségünk lesz a további tűzfal opciók beállítására. Fontos, hogy nem lesz a rend-

szer indulásakor aktív, a /proc/sys segítségével kell bekapcsolni:
echo „1" > /proc/sys/net/ipv4/ip_forward
Nem részletezünk minden opciót, mivel véleményünk szerint egy komoly tűzfal építéséhez komoly szakértelem is kell, így amit mi itt el tudnánk mondani nagyon kevés. Javasoljuk elolvasásra a FIREWALL-HOWTO dokumetumot, és sok irodalmat a tűzfalakkal kapcsolatban. Egyszerű IP-maszkolást végző rendszert a Magazinban leírtak szerint lehet építeni. Figyeljünk oda, hogy az IP-maszkolás működéséhez szükség van az „always defragment" opcióra is, mely a több darabban érkező IP-csomagok összeállítását jelenti. Itt kapcsolhatjuk be a transzparens proxy funkciókat is.

Rendes körülmények között olyan technikát alkalmaz a kernel az IP-csomagok kezelésekor, mely egy útválasztóban nem igazán előnyös. Ilyenkor érdemes bekapcsolni az „optimize as router not host" opciót. Ha nem útválasztónak használjuk a gépet, kapcsoljuk ki. Ritkán használt funkció a „tunneling", mely elvileg egy adott protokoll beágyazása egy másikba, de jelen esetben csak IP-t ágyazhatunk be IP-be. Ez elsőre furán hangzik, de nagyon jól használható mobil gépekhez, ha azt szeretnénk, hogy állandóan fix IP-számmal egy adott hálózathoz csatlakozzon a gép. Többnyire erre nincs szükség, azaz kapcsoljuk ki.

Egy hasonló beágyazási módszer a GRE is, melyet például a Cisco útválasztók is ismernek, azaz jobban használhatjuk a velük folytatott kommunikációban, mint az előzőt. A GRE segítségével IPv4-et és IPv6-ot is beágyazhatunk IPv4 csomagokba, azaz összeköthetünk két IPv6 hálózatot IPv4-et ismerő csatornán. További számos, útválasztókat üzemeltetők számára fontos opció következik, melyekre nem térünk ki részletesen. A kernel dokumentációja foglalkozik velük.

Előfordul, hogy egyetlen hálózati csatolónknak valamilyen okból több IP-számot szeretnénk adni. Ezt teszi lehetővé az „aliasing support". Így az alap eth0 (első ethernet csatoló esetén) név mellett használhatjuk az eth0:0, eth0:1 stb. neveket is, mindnek különböző konfigurációt adva meg, miközben ugyanazt a fizikai csatolót használják. Ismert támadási forma az Interneten az úgynevezett SYN flood, amikor folyamatosan küldünk kapcsolódási kérelmeket egy szervernek, de választ nem várunk. Ha kellően gyorsan küldjük ezeket, akkor túlterheljük a szervert, ami így képtelenné válik más kapcsolat fogadására, amíg abba nem hagyjuk a támadást. Ezt hívják DOS támadásnak (Denial of Service Attack). Ez ellen lehet védekezni a „SYN cookies" segítségével, mely ha észreveszi, hogy gyorsan jönnek kapcsolati kérelmek a másik géptől, egy – SYN cookies nevű – speciális protokollt kezd használni, annak érdekében, hogy aki jogosult bejuthasson, a támadó pedig ne járjon sikerrel. Fontos, hogy nem elég csak itt bekapcsolni, a működő rendszeren élesíteni is kell a /proc/sys-en keresztül:
echo 1 >/proc/sys/net/ipv4/tcp_syncookies
Érdemes bekapcsolni.

Ha diskless, azaz lemez nélküli terminálokat szeretnénk a hálózaton üzemeltetni, kernelszinten két módszerrel adhatunk nekik IP-számot és egyéb hálózati konfigurációt: az egyik a felhasználói szinten futó bootp protokoll, a másik a RARP. Ha ez utóbbit kívánjuk használni, itt be kell kapcsolni a RARP támogatást (Reverse ARP). Nagy sebességű és nagy távolságú kapcsolatoknál csökkentheti a sebességet, ha a küldő gép kevés adatot tud pufferelni. Ha az „Allow large windows" opciót bekapcsoljuk, ez a mennyiség jelentősen nagyobb lehet. 16 megabyte vagy kevesebb memóriával rendelkező gép esetén ezt nem ajánljuk, mert egy-egy kapcsolat igen sok memóriát felemészthet ilyenkor. Jelentősége a beállításnak csak 2 Mbit/s sávszélesség felett van.

Mint tudjuk, fogytán vannak az IP-számok a Neten. Többek között ezt a problémát is megoldja az IPv6 nevű új Internet protokoll verzió. Ezt még hivatalosan nem vezették be, de folyik a fejlesztése. Szerencsére, a Linux kernel már támogatja az IPv6-ot, így akár útválasztóként, akár egyszerű munkaállomásként kapcsolódhatunk Linuxszal ilyen hálózathoz. Mivel használatát csak szakembereknek javasoljuk, az opcióira itt nem térünk ki bővebben.

Az IP mellett a Novell rendszerek jóvoltából igen elterjedt kommunikációs protokoll az IPX és az SPX. Ezeket használják a Novell szerverek és a kliensek a kötetek eléréséhez. Természetesen a Linux sem maradhat ki ebből a világból, így megvan a lehetőségünk az IPX és az SPX használatára is. Ezek futás közbeni konfigurációja teljesen automatizálható, így ha bekapcsoltuk, sokat nem kell vele törődni (persze az IPX beállító programokat fel kell tenni). Ha Netware klienseket szeretnénk kiszolgálni akár a Mars Netware emulátorral, akár linuxos Novell 4.1-gyel, szükségünk lesz az IPX-re. Modulba érdemes tenni, így csak akkor foglal memóriát, ha tényleg használjuk. Ugyanezt tegyük az SPX-szel, ha szükségünk van rá.

A világ másik felén sokan szeretik a Macintosh rendszereket. Ez tényleg egy másik világ – saját szabványokkal, saját protokollokkal. Ki mondta, hogy a Linux nem ért velük szót? Kapcsoljuk be (inkább tegyük modulba) az „Appletalk DDP" funkciót, és máris megvan a szükséges kerneltámogatás. Ha feltesszük a netatalk csomagot, rögvest file- és nyomtatószerverként üzemel a Linuxunk MacOS-t futtató gépek számára. Mondtuk, hogy a Linux mindenkivel beszélő viszonyban áll, még-

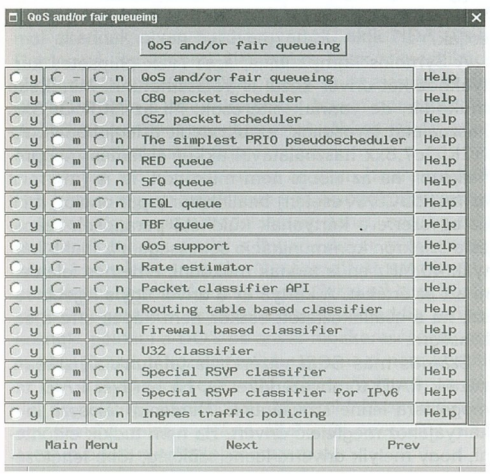
is mi a helyzet a DECnettel és az SNA-val? Bár még erősen fejlesztői stádiumban, de a Neten már találunk DECnet drivert Linuxhoz, és az IBM valahol ígérte az SNA támogatást is.

Visszatérve a kernelbeállítás rejtelmeihez, a hálózati résznél találunk még pár, jórészt kísérleti funkciót. Ami említést érdemel közülük, az a CCITT X.25, mely az X.25 hálózaton történő kommunikációt teszi lehetővé, valamint a Bridging, amivel az ethernet hálónkon bridge-nek használhatjuk a Linux szervert, azaz látszólagosan egy subnetté alakíthatunk fizikailag kettőt vagy többet. Az útválasztáshoz tartozik a WAN router opció. Ez X.25, frame relay és bérelt vonali kapcsolatoknál használatos. Ha igen gyors kapcsolásra van szükségünk, kapcsoljuk be a „Fast switching" opciót. De vigyázat, ez nem kompatibilis a tűzfal funkciókkal, hiszen (a sebesség kedvéért) bizonyos kódokat megkerülve viszi át az adatokat a két hálózati adapter között. Általában sem erre, sem az előző opciókra nincs szükség, csak az kapcsolja be, aki pontosan tudja mit akar velük. Még egy érdekes opció, a „CPU is too slow to handle full bandwith". Ha a processzorunk nem elég gyors, próbáljuk meg bekapcsolni ezt, hátha gyorsít valamit a hálózati funkciókon.

Csomagütemezés

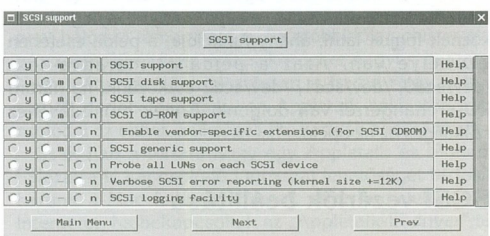
Még mindig a hálózati beállításoknál tartunk, de most egy igen speciális funkció beállítása következik. Ha ezekre nincs szükségünk, márpedig egy átlagos rendszerben nincs, kapcsoljuk ki a „QoS and/or fair queing" opciót, és lépjünk tovább a következő menübe.

Ha szükségünk van arra, hogy különféle algoritmusokat használjunk annak eldöntésére, hogy egy-egy csomag mikor kerüljön továbbításra, kapcsoljuk be a QoS funkciót. Erre olyan esetekben lehet szükség, ha valósidejű kapcsolatot használunk, és garantálni kell egy bizonyos minimális átviteli sebességet. Természetesen van egy alapalgoritmus ennek eldöntésére (ez fog működni, ha kikapcsoljuk a QoS-t), de sokszor ez nem elegendő. Használatát csak profiknak ajánljuk egyelőre, ugyanis pontos ismeretekkel kell rendelkezni a hálózati kommunikációról, hogy helyes döntésekkel befolyásolhassuk a működést. A használatához szükséges programot letölthetjük a ftp://ftp.inr.ac.ru/ip-routing/ FTP-címről. Ha bekapcsoltuk, a /proc/net/psched fileban olvashatjuk az aktuális állapotot.



SCSI támogatás

A SCSI rendszer a nagygépes rendszerekből származik, és egy kivételesen igen szabványos protokoll a gép és perifériái kommunikációjára. Emellett azt is elmondhatjuk róla, hogy egy nagyon megbízható rendszer. A SCSI vezérlő sok funkciót levesz a processzor válláról, ezáltal csökkenti annak terhelését, és növeli a rendszer teljesítőképességét. Példának okáért a diszk I/O jó részét a SCSI vezérlő végzi, ellenben az IDE/ATAPI rendszerekkel, ahol a CPU-t terheli ez is. Természetesen ennek a tudásnak a szó szoros értelmében ára van, és ez a legnagyobb érv a SCSI eszközök ellen. Egy SCSI merevlemez általában másfél–kétszer annyiba kerül, mint egy azonos kapacitású IDE diszk. Ennek ellenére szerverekbe, nagy terhelésű rendszerekbe, vagy ahol a 24 órás rendelkezésre állás alapvető követelmény mindenképpen érdemes ezt használni.



A Linux kernel szinte minden SCSI funkciót támogat, illetve találunk külső modulokat bizonyos speciális eszközökhöz is, mint például a SCSI DAT Changer (itt a

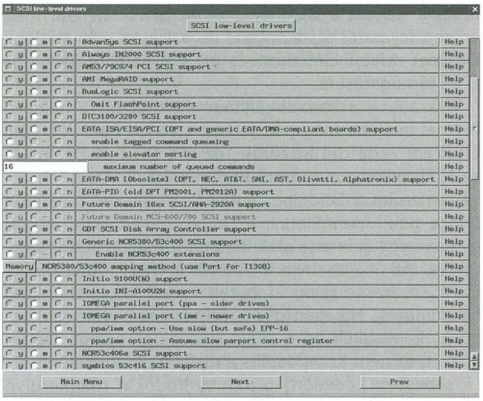
DAT-tól különállóan építettre gondolunk). Fontos megjegyezni, hogy például az Iomega ZIP drive párhuzamos portra csatlakoztatható változata is a SCSI protokollt használja a kommunikációra egy speciális driver segítségével, így ha amúgy nem lenne SCSI eszközünk, ehhez szükséges a SCSI támogatás. Mi azt javasoljuk, hogy mindenki legalább modulba tegye be az összes SCSI funkciót, hiszen így nem sok helyet foglal, viszont bármikor felhasználható, ha kell. Természetesen, ha a rendszer SCSI diszkről bootol, a SCSI és a SCSI diszk támogatásokat a kernelbe kell fordítani.

A „tape", CD-ROM és „generic" támogatásra legritkább esetben van szükség a rendszer indításakor, így azokat célszerű modulba tenni. A „generic" támogatást használják például a CD-írók, így ha CD-írásra adtuk a fejünket, erre szükség lesz. Itt kell visszautalnunk egy, még az IDE beállításoknál található funkcióra, a SCSI emulációra. Ez egy olyan trükk, mellyel IDE eszközt a SCSI protokoll segítségével üzemeltethetünk. Természetesen ez nem adja a SCSI tényleges előnyeit, de néha szükséges lehet. Ilyen eset jópár ATAPI CD-íróé, melyeket a SCSI CD-írókéval azonos módon kellene kezelni. Ahhoz, hogy ezt Linux alatt megtehessük, szükség van a SCSI-emulációra IDE buszon, a SCSI és a SCSI CD-ROM és generic támogatásra. A CD-ROM támogatás azért kell, mert a SCSI emuláció miatt ATAPI CD-ROM-ként nem látjuk többet, így azt SCSI-ként kell kezelni. Értelemszerűen az IDE/ATAPI CD-ROM támogatás ilyenkor nem lehet a kernelben (ha az modulban volt, egyszerűen el kell távolítani a modult a memóriából), hiszen az előbb megtalálja az egységet, mint betöltenénk a SCSI emulációt, és az utóbbi nem fog működni.

Az alap-SCSI beállításoknál még egy opcióval érdemes foglalkoznunk, a „Probe all LUNs on each SCSI device"-szal. Van, hogy egy SCSI ID-vel rendelkező eszközt több eszközként kell látnunk. Erre jó példa a CD-changer, azaz a több CD-t kezelni tudó CD-ROM. Ilyenkor az eszköz több úgynevezett LUN-t (Logical Unit Number) ad vissza, mellyel annyi tényleges eszköznek fogjuk látni, ahány funkciója, a példa esetében CD-helye van. Azaz a példánál maradva lesz /dev/scd0, /dev/scd1, /dev/scd2 eszközünk, ha három CD-s changerrel van dolgunk. Ennek működéséhez szükséges ez az opció.

SCSI vezérlők beállítása

Az előzőekben csak az általános SCSI támogatást állíthattuk be, de természetesen minden SCSI kártyának megvan a saját drivere, és a működéshez erre is szükség van.



Itt is abban a szerencsés helyzetben vagyunk, hogy mindent tehetünk modulba. Érdemes is így tenni, legalábbis ha szándékunkban áll többféle SCSI vezérlőt is kipróbálni, vagy a kernelünket hordozni olyan gépek között, ahol eltérő SCSI vezérlők vannak. Természetesen azon vezérlő driverét, amelyről a rendszer bootol, bele kell fordítani a kernelbe (vagy használjunk initrd-t, de ez utóbbi sokkal bonyolultabb megoldás). Minden elterjedt SCSI vezérlőhöz találunk itt drivert. Ezek közül talán az Adaptec és a Symbios kompatibilis kártyákból van a legtöbb a gépekben. Az előbbihez az AIC7xxx driver kell (kivéve a régebbieket, melyekhez van egyedi driver, mint például az AHA1542). Ha modulba tettük vagy bekapcsoltuk az AIC7xxx támogatását, beállíthatunk pár további opciót is, melyek a működést befolyásolják.

A Symbios kompatibilis, azaz NCR chipsetes kártyákhoz két driver közül is választhatunk. Mindkettő működik, de vannak eltérések. Mi az NCR53C8XX használatát javasoljuk, ez bizonyult a legjobbnak. Az NCR53c7,8xx használatával akkor érdemes megpróbálkozni, ha az előbbi nem működött. Itt is (mármint az NCR53C8XX esetén) beállíthatunk pár opciót, mint az egyszerre a kártyának küldhető parancsok száma és a szinkron kommunikáció sebessége. Az újabb kártyák 40 MHz-en is tudnak működni, érdemes felemelni ezt az értéket. A kártya és a driver úgyis lejjebb veszi, ha a csatlakoztatott eszköz nem képes erre a sebességre.

Számos más SCSI vezérlőt is találunk a listában, köztük RAID tömbvezérlőket is, ha hardver RAID támogatásra lenne szükségünk. Mindenki válassza ki a kártyájához megfelelő drivert. Ha nem tudjuk eldönteni, hogy melyik driverre lenne szükség, több lehetősé-

günk is akad. Az egyik, hogy a Debian telepítővel indítjuk a gépet, melybe a SCSI driverek bele vannak építve, és megnézzük, minek ismeri fel a kártyánkat a kernel. A másik, hogy IDE merevlemez segítségével elindítunk egy Linuxot az adott gépen, felteszünk rá egy olyan kernelt, melybe az összes SCSI vezérlő modulba van fordítva, majd kézzel a modprobe parancscsal egyenként végigpróbáljuk azokat. Ez a trükk beválik például a hálózati kártyáknál is, de erről majd később.

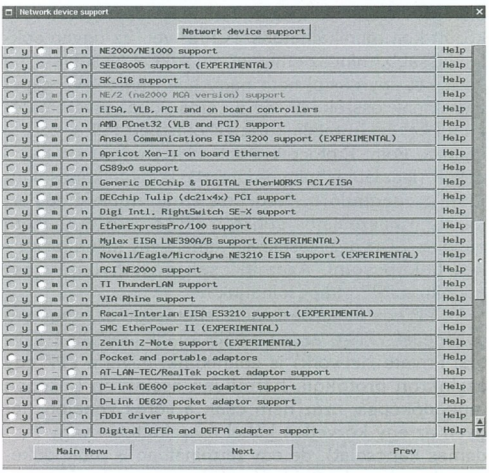
Itt találjuk a már többször említett Iomega ZIP párhuzamos portos verziójának a driverét is, azonnal kettőt is belőle. Az egyik PPA, a másik IMM névre hallgat. A régebbi, csak párhuzamos porton működő verziókhoz a PPA, az újabb, párhuzamos porton és SCSI-n is működő verzióhoz az IMM driver kell. Ez utóbbi a ZIP Plus névre hallgat, és a kábelen „Autodetect" felirat olvasható. Érdemes mindkettőt modulba tenni, ki tudja mikor állítanak be hozzánk egy ilyennel.

A hálózati eszközök beállítása

Természetesen a többi hardvereszközhöz hasonlóan a hálózati eszköz driverek is a kernel részét képezik. Igaz, napjainkban egyre több hardvergyártó támogatja hivatalosan a Linuxot, de ez többnyire csak a hardverspecifikációk közreadását jelenti, ami köztünk szólva jobb, mint a más rendszereken szokásos, az eszközhöz adott hol jó hol nem jó driverek. Ugyanis ez azt eredményezi, hogy olyan profi linuxos hackerek írják meg a drivert, akik a lehető legtöbbet akarják kihozni az eszközből, így a driver tényleg nagyon jó lesz. Sajnos, szép számmal akadnak olyan hardvergyártók, akik NDA (Non-Disclosure Agreement) aláírásához kötik a specifikációk kiadását, azaz megtiltják, hogy az az alapján írt driver bekerüljön a Linux kernelbe, hiszen oda csak forráskódban kerülhet be, de az NDA többek között a forráskód publikálását is tiltja.

Őszintén szólva nem értem az ilyen hardvergyártókat, hiszen ingyen adják más operációs rendszerekhez a drivert, így azon nem keresnek pénzt, míg ha több embernek tudnák eladni a linuxos driver miatt az eszközt, azon keresnének. Mindemellett sok ilyen – mondhatni nem túl kedves – cég élvezi a Linux közösség támogatását, ugyanis a hackerek addig próbálkoznak, például a DOS-os driver hibakeresésével, míg rá nem jönnek a kezelés mikéntjére, és el nem készítik az így szabad szoftverré váló drivert. Volt már rá példa, hogy az így összehozott driverrel jobban működött egy eszköz, mint a gyártó által más operációs rendszerhez adott meghajtóprogrammal.

Visszatérve a hálózati eszközökre, ha belépünk a „Network device support" menübe, egy igen hosszú lista tárul elénk.



Természetesen itt is először azt kell eldönteni, akarjuk-e a hálózati eszköz támogatást a kernelbe. Ha úgy gondoljuk, sosem fogjuk gépünket hálózatba kötni, kapcsoljuk ki. Mi azért javasoljuk a hálózati támogatás bekapcsolását, még akkor is, ha most úgy tűnik sosem lesz rá szükség. Mivel ide tartozik a PPP is, akkor is kapcsoljuk be, ha modemes Internet-elérést szeretnénk használni.

A PPP mellett megtaláljuk itt a ma már kissé ódivatú ARCnet támogatást, az ethernet és számos ethernet kártya támogatását. Ugyancsak lelhetünk FDDI (Fiber Distributed Data Interface) és a HIPPI (HIgh Performance Paralell Interface, egy 800 megabit/másodperc és 1600 megabit/másodperc sebességű switchelt pont-pont hálózat) támogatást is. Ez utóbbi főleg cluster alapú szuperszámítógép építésekor lehet nagyon hasznos. Nem hiányozhat természetesen a frame relay sem a sorból, de lehetőségünk van Appletalk hálózatba kapcsolódni az ehhez szükséges PC-kártyák segítségével.

Csak hab a tortán, hogy egy Appletalk és IP alapú hálózatra is kapcsolódó Linuxszal lehetőséget biztosíthatunk a Mac felhasználóknak arra, hogy elérjék az Internetet, ha bekapcsoljuk az „Appletalk-IP driver" támogatást, valamint az „Appletalk-IP to IP Decapsulation" támogatását. Természetesen mehet a dolog fordítva is, az „IP to Appletalk-IP Encapsulation" segítségével IP-csomagokat is továbbíthatunk Appletalk háló-

zaton, amit remélhetőleg valaki valahol (például egy másik Linux) majd visszaalakít „rendes" IP-csomaggá.

A PLIP egy a PPP-hez hasonló pont-pont összeköttetés, csak éppen a párhuzamos portot használja. Segítségével két, párhuzamos kábellel összekötött gép között kommunikálhatunk TCP/IP-vel. Ez ugyan sokkal lassabb, mint az ethernet, de sokkal olcsóbb is, ha csak két egymástól nem nagy távolságra levő gépet szeretnénk összekötni, de a soros vonal sávszélessége túl kevés. Itt találjuk a már jól ismert PPP-t, azaz a Point-to-Point Protocollt. Mint az köztudott, ezt használjuk (legalábbis túlnyomórészt), hogy felkapcsolódjunk a világhálóra. Meg kell említenünk, hogy az ISDN-en használatos szinkron PPP-t nem itt, hanem majd az ISDN beállításoknál – amely egy külön menü – kell bekapcsolni. Azaz itt csak a „hagyományos" aszinkron PPP-t kapcsolhatjuk be.

A PPP egy kevésbé ismert lehetősége az, hogy segítségével VPN-t, azaz virtuális magánhálózatot (Virtual Private Network) is építhetünk. Mivel a PPP egy aszinkron protokoll, bármilyen aszinkron közegen működőképes, azaz nem csak a soros vonalon (legyen az egy közvetlen kábel összeköttetés vagy modemes kapcsolat), de egy terminálban is működőképes. Azaz ha telnettel belépnénk egy másik gépre, ott elindítanánk a ppp-t, hogy az a terminált használja kommunikációs csatornának, és a kiinduló gépen egy helyi PPP-be tudnánk ezt az adatot irányítani, illetve a helyi PPP kimenetét a telnet túloldalán lévő PPP programba, máris létrejönne egy IP-kapcsolat. Ezt nevezzük VPN-nek, de ez így alapesetben nem megy, mivel a telnet program ki- és bemenetét nem tudjuk megfelelően átirányítani, és sok értelme sincs, hiszen az adataink így is kódolatlanul mennek, pedig a védelem az egyik leghasznosabb lehetőség egy VPN építésekor. Nem kell csüggedni, van megoldás, mégpedig az ssh, mely képes a ki- és bemenetét programba irányítani az rsh-hoz hasonlóan, ráadásul még kódolja is az adatokat. A megoldás ugyan működik (a megvalósítás leírását megtaláljuk a VPN-HOWTO-ban), de kellemetlen szépséghibája, hogy mindkét gépen be kell lépni hozzá egy speciális felhasználóval, és az, hogy az ssh program nem ingyenes.

Egy kevés programozást igénylő módszer, ha SSL-t (Secure Socket Layer) használunk, mely egy igen megbízható kódolt kommunikációs felületet biztosít, és van ezt megvalósító szabad szoftver. Azért szükség van némi programozásra, mivel az eredeti PPP program terminálon keresztüli működésre lett kitalálva (a soros port is egy terminál ebben az értelemben), és emiatt bizonyos elvárásai vannak. Azaz módosítani kell a természetesen szabad szoftver pppd forrását, hogy megfelelő opcióval hajlandó legyen az stdio-ján keresztül bárkivel kommunikálni anélkül, hogy ott terminált keresne. Így azonnal két legyet üthetünk egy csapásra: nem kell bejelentkezni a másik gépre, a jogosultság ellenőrzését egyrészt az SSL szintjén, másrészt a PPP azonosítási szintjén elvégezhetjük, és a megoldás teljes egészében szabad szoftveren alapul.

Ennyi kitérő után térjünk vissza a hálózati eszközökhöz. A PPP-nél hagytuk abba, így folytassuk annak többé-kevésbé elődjénél, a SLIP-nél, azaz a Serial Line Internet Protocollnál. Ez egy egyszerűbb, butább megvalósítása a soros vonalon történő internetezésnek. Az Internet ilyen mértékű térhódítása előtt ez volt a legelterjedtebb protokoll, ha soros vonalon, azaz többnyire modemen kellett két gép között IP-kapcsolatot létesíteni. Ma már átvette a helyét a PPP, mivel a SLIP nem tud autentikációt, és nem képes átadni a hálózati beállítási adatokat a túloldalnak. A SLIP kapcsolatok beállításakor előre meg kellett adni valamennyi beállítást. Ez nem volt túl kényelmes, hatékony avagy biztonságos. Persze akkor, amikor ezt kitalálták, ez nem is volt igény, ahogy az IP kódolatlansága is ezen igény hiányából eredt, hiszen akkor annyira zárt rendszerről volt szó, hogy nem volt ki ellen védekezni.

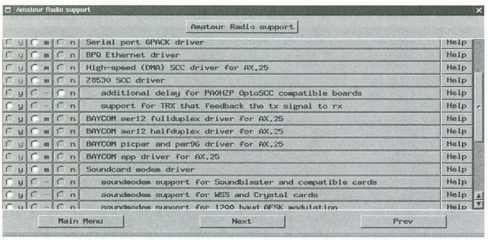
Mára ez gyökeresen megváltozott, és a már említett IPv6-ban a kódolt kommunikációt is megvalósították. Ha szükségünk van a SLIP-re – márpedig ha a diald programot szeretnénk használni szükségünk lesz rá –, mivel az egy virtuális SLIP eszközt hoz létre a működéséhez; tegyük legalább modulba. Kernelbe fordítani teljesen felesleges, kivéve ha egy SLIP-en keresztüli hálózati kapcsolatot kívánunk felhasználni a rendszer elindítására, mert ezt is lehet (a PPP-vel is lehet természetesen). Találunk még pár SLIP opciót alatta, melyeket érdemes bekapcsolni. Ezek a CSLIP, mely a SLIP csomag fejlécek tömörítését szolgálja, a „Keepalive and linefill" rossz minőségű vonalak esetén lehet hasznos, míg a „Six bit SLIP encapsulation" akkor, ha valamiért a szokásos 8 bites kapcsolat helyett 6 biten el kell férnie az adatoknak. Erre egy jellemző alkalmazási forma, amikor valahol van egy terminálos behívópontunk, de nincs elvileg lehetőségünk SLIP vagy PPP kapcsolatra. Ilyenkor a Slirp programmal és a linuxos SLIP-pel mégis tudunk internetezni. De mivel az ilyen terminálkapcsolatok jellemzően maximum 7 bitesek, szükség van erre az opcióra a működtetéshez.

Találunk a kernelben rádiós (itt nem amatőr rádióról van szó) kapcsolat támogatást, Tokenring támogatást, WAN (Wide Area Network), valamint aszink-

ron kapcsolaton működtethető X.25 támogatást is. Egy igen hasznos funkció még a „Traffic Shaper", mely segítségével szabályozhatjuk egy adott eszközön maximálisan kimenő adatok mennyiségét.

Amatőr rádiózás

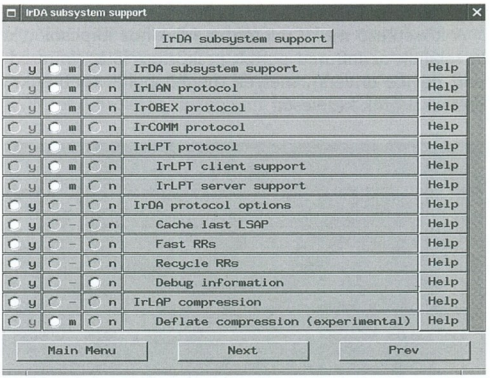
A hálózatoknál már volt szó a rádiós kapcsolatról, de ott még nem az amatőr rádiózásról. Az amatőr rádiózás nagy előnye az alacsony költség. A szükséges berendezések megfizethetők, a kommunikáció ingyenes, igaz a sebesség igen alacsony.



Az „Amateur Radio support" menüben az ehhez szükséges funkciókat kapcsolhatjuk be. A megszokott amatőr rádiós technikákon kívül találunk itt egy igen érdekeset, mely egy sima SoundBlaster vagy azzal kompatibilis hangkártyát használ az adatkommunikációra a rádión keresztül. Telefonvonalon ez nem működik, de amatőr rádiózás esetén megspórolhatjuk a rádiós modemet ha van egy hangkártyánk, és akár a 9600 baudos sebességet is elérhetjük. Ez a technika egy 75 MHz-es Pentium processzor teljesítményének négy százalékát igényli.

IrDA

Egyre divatosabb a kábeles kommunikáció helyett a vezeték nélküli kapcsolat. Ennek egyik megvalósítása az infravörös csatlakozás, melyet bizonyára mindenki ismer, akinek volt már a kezében tévé vagy hifi távirányító. Ugyanezt a technikát másképp is alkalmazhatjuk. Erre a vivőre építve fejlesztették ki az IrDA (Infrared Data Associations (tm)) protokollokat, melyeket leggyakrabban hordozható számítógépeken (notebook, laptop és PDA) alkalmaznak, de asztali PC-khez is találunk megfelelő hardver eszközt e célra. Az IrDA protokolljai számos kommunikációs formát ismernek, és ha a kernelben bekapcsoljuk az általános támogatást, lehetőségünk nyílik ezeket egyenként ki- vagy bekapcsolni.



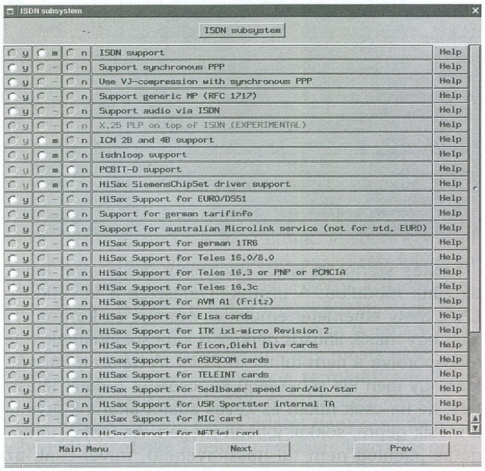
Szokás szerint itt is azt javasoljuk, hogy amit lehet, tegyünk modulba. A protokollválaszték a következő: LAN, mely ethernetet emulál és számos eszközhöz, de akár egy másik linuxos géphez is kapcsolódhatunk vele megspórolva a nem túl olcsó PCMCIA hálózati kártyát. Igaz, így igen kis sebességű kapcsolatot tudunk létesíteni, de ha csak ritkán és kevés adatot akarunk átvinni a másik gépre, sokkal jobban megéri. Az OBEX támogatás jelenleg nem túl hasznavehető, mivel csak az alapvető dolgokat valósítja meg. Talán a legsokoldalúbban használható IrDA protokoll az IrCOMM, mely egy soros kapcsolatot emulál, így ezen bármilyen más, soros portot használó kommunikációs rendszert használhatunk, ideértve a PPP-t is (igaz TCP/IP-hez az IrLAN jobb, ha mindkét oldal támogatja). Az IrCOMM használható IrDA kompatibilis rádiótelefonokkal történő modemezéshez és adatcseréhez is.

Néhány nyomtató is ismeri az IrDA protokollt, és ha bekapcsoljuk az IrLPT funkciót az „IrLPT client" támogatással együtt, azonnal nyomtathatunk is az ilyen nyomtatókra. Ezt elsősorban a laptop tulajdonosok fogadhatják örömmel. Sokkal érdekesebb lehetőség azonban, ha bekapcsoljuk a szerverünkön az IrLPT mellé az „IrLPT server" támogatást is. Ekkor a Linux szerverünk egy IrDA kompatibilis nyomtatószerverré válik, melyre bármilyen IrDA-s kliensről, tehát az előbb említett noteszgépről, vagy PDA-ról azonnal nyomtathatunk, és nincs szükség IrDA-t ismerő nyomtatóra.

Az utolsó opció (IrLAP), egy linuxos fejlesztés, és nincs benne az IrDA szabványokban. Ez egy tömörítő algoritmus, mely a gzip-en alapul, hasonlóan a ppp tömörítéshez. Ha bekapcsoljuk, és mindkét oldal támo-

gatja, kellemesen felgyorsíthatja az átvitelt. Jelenleg csak két Linux rendszer között működik, de mivel nyílt forrású kódról van szó, semmi akadálya nem lehet elterjedésének.

A következő menü szintén az IrDA-hoz kapcsolódik. Itt találjuk az infrared portok eszközkezelőjét (Infrared-port device drivers), melyek megkönnyítik számunkra ezen eszközök kezelését.

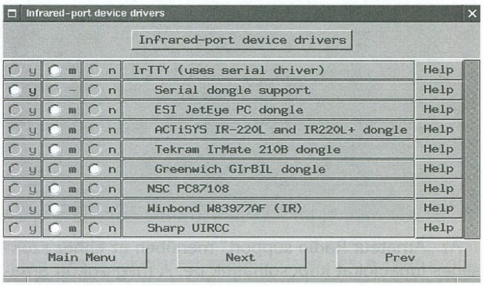


Modulba érdemes ezt is tenni, hiszen nem minden esetben van szükség rá. Az IrTTY bekapcsolása (illetve a moduljának betöltése) a soros port-kompatibilis IrDA eszközökhöz jó, melyek ilyenkor IrDA SIR módban működnek, és kezelhetők a Linux szabványos soros portkezelő programjaival, hiszen egy 16550 kompatibilis soros portot kapunk. Természetesen ez a mód korlátozza a sebességet a 16550-es soros port maximális 115200 bit/s teljesítményére. Az „NSC", „Winbond" és „Sharp" opciók egy-egy eszköz kezelőprogramjait kapcsolják be. Szintén tegyük modulba, és majd a futó rendszeren töltsük be a megfelelő eszköz driverét, ha rendelkezünk ilyennel.

ISDN alrendszer

ISDN, azaz Integrated Services Digital Network. Napjaink egyik divatos kapcsoltvonali kommunikációs rendszere. Magyarországon fő jellemzője a magas ár, a világ fejlettebb részein a nagy elterjedtség. Hatalmas előnye az analóg telefonrendszerrel szemben a nagy megbízhatóságú kapcsolat, a gyors kapcsolás és a nagy átviteli sebesség. Ha abban a szerencsés helyzetben vagyunk, hogy ki tudtunk fizetni százezer forintot egy ilyen vonalért, vagy cégünk tette ezt meg, és ránk hárul a feladat, hogy megoldjuk rajta keresztül az Internet-elérést, szokás szerint a Linux a megfelelő választás, és itt kell bekapcsolnunk a kernel szintű támogatást hozzá. Természetesen emellett még szükség lesz pár programra, de ezek is megtalálhatók a CD-ken isdnutils néven.

Talán mondanunk sem kell, hogy modulba tegyük, hiszen ritkán van szükség arra hogy ISDN eszközről bootoljon a gép. Ha bekapcsoltuk az ISDN támogatást, további számos opció lesz elérhető.



Itt találjuk a szinkron PPP támogatást. Amint azt az aszinkron PPP-nél írtuk, az ISDN-hez szinkron PPP kell, mivel az analóg modemes kapcsolatokkal szemben az ISDN szinkron kapcsolatot használ. Értelemszerűen másik PPP program is kell, de ez benne van a már említett isdnutils csomagban. Természetesen megoldható az aszinkron PPP is, hiszen egy terminálkapcsolat is létesíthető két ISDN állomás között, csak értelme nincs sok, hiszen a szinkron PPP gyorsabb.

Tökéletes. Van ISDN-ünk, megy a szinkron PPP, száguldozhatunk a szupersztrádán akár 128 kbit/s-mal is (ISDN2 csatlakozás esetén). De ugye a telefont nem csak internetezésre akarjuk használni, szoktunk rajta beszélni is. Mivel ezt az ISDN-en is meg lehet tenni, nem kell külön analóg vonalat venni mellé. De ha van számunk, ahol hívhatnak minket, kell legyen üzenetrögzítő is, hátha nem vagyunk ott. Az ISDN készülékeket is a magas ár jellemzi, így jobb elkerülni annak megvételét, és szerencsénkre a Linux kernel támogatja a hangkapcsolatot az ISDN-en keresztül. Ha bekapcsoljuk a „Support audio via ISDN" opciót, lehetőségünk van az aszinkron modemeknél megszokott „voice" parancsokkal kezelni az eszközt, azaz elindíthatunk például egy vgetty-t, mely kiválóan képes az üzenetrögzítői

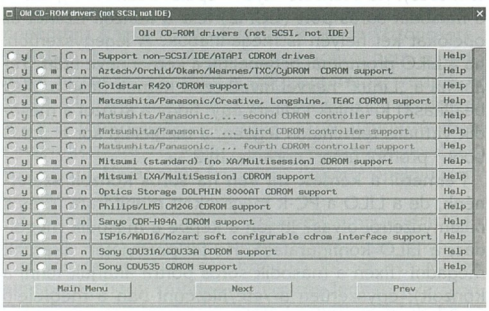
funkciókat ellátni. Jelenleg ezt a funkciót csak a HiSax driver tudja, azaz csak olyan ISDN kártyákkal működik, melyeket ezzel a driverrel kezelünk. Alább látni fogjuk, hogy a kártyák többsége ilyen.

Az „X.25 PLP" opcióval akár X.25 protokollal is beszélgethetünk egy ISDN kapcsolaton keresztül. A továbbiakban különböző ISDN kártyák támogatását lehet beállítani, kivéve az „isdnloop" opciót, melyet a kernel fejlesztői a pénztárcánk megkímélése érdekében tettek a kernelbe. Segítségével konfigurálhatunk egy virtuális ISDN eszközt, melyen tesztelhetjük a beállításokat anélkül, hogy közben a hívásokért számlázna nekünk a szolgáltató.

Mielőtt ISDN kártyát vennénk, érdemes alaposan tanulmányozni a Linux által támogatott ISDN kártyák listáját, nehogy pont egy nem támogatott kártyával lepjük meg számítógépünket. A kernel beállításnál látható lista és help információk mellett érdemes belekukkantani a kernel forráskódjába, azon belül a Documentation/isdn alkönyvtárba, ahol számos dokumentumot olvashatunk az ISDN-nel kapcsolatban.

Nosztalgikus CD-ROM-ok

Bizonyára sokan emlékeznek a PC-be tehető CD-ROM-ok hőskoráról, amikor egy aranyos pici kártyát is kaptunk sok pénzért beszerzett eszközünkhöz, vagy legalábbis egy hangkártya kellett, melyhez csatlakoztattuk. Szerencsére ez ma már a múlté, hiszen minden új CD-ROM vagy SCSI vagy IDE/ATAPI interfésszel rendelkezik. Ennek ellenére talán még vannak olyanok, akiknek megfelel a régi, egy- vagy kétszeres sebességű CD-ROM-juk, és nem akarnak újat venni. A Linux kernel már régóta tartalmazza ezen eszközök kezelőprogramjait. Az „Old CD-ROM drivers" menüben tudjuk bekapcsolni őket. Ha nincs ilyen eszközünk, és nem is szándékozunk az őszerest felkeresni érte, egyszerűen kapcsoljuk ki ezt a támogatást és lépjünk tovább. Ha van ilyen eszközünk, kapcsoljuk be. Az első pont („Support non-SCSI/IDE/ATAPI CDROM drives") csak egy elvi választás, ezért nincs értelme modulba tenni, de az alatta következő opciók már a tényleges kezelőprogramokat kapcsolják ki és be. A modernebb CD-ROM eszközökhöz hasonlóan javasoljuk, tegyék ezt is modulba.

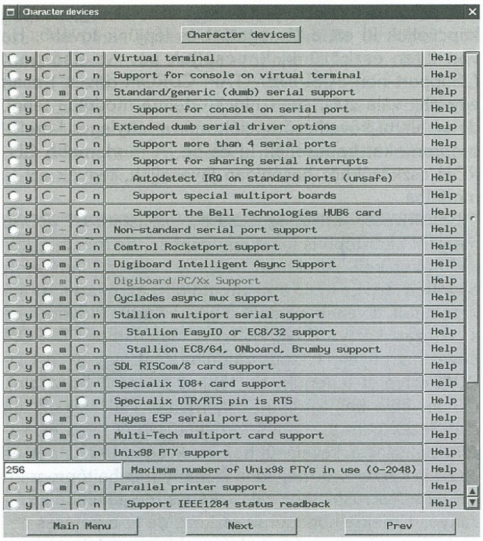


Egy kivétel van, méghozzá a Matsushita CD-ROM, melyből akár több is lehet egy gépben, de mindhez külön driver kell. Általában az eszközökhöz Linux alatt egyetlen driver kell, ami megtalálja az összes általa kezelhetőt. Ilyenek például a SCSI kártya driverek és a hálózati kártya driverek is. A Matsushita CD-ROM driver nem ilyen. Viszont modulba csak egy ilyen CD-ROM driverét tehetjük. Ha többet akarunk használni, mindet bele kell fordítani a kernelbe. Ne felejtsük el, hogy ezek nem a könnyű kezelhetőségről híres eszközök, sokszor a kezelésükhöz szükséges I/O címet és IRQ-t is kézzel kell megadni. Néhány meghajtóprogram – mint a Matsushitához való – képes megtalálni a szokványos címekre konfigurált eszközt, de ez a keresés néha percekig is eltart, így érdemesebb megadni a pontos adatokat.

Karakteres eszközök

Talán érdemes pár szót szólni arról, mi is az a karakteres eszköz. A Linux alatt, mint a telepítőben már leírtuk, a /dev könyvtárban találjuk a legtöbb hardver eléréséhez szükséges file-t (néhányat a /proc alól lehet kezelni). Ezek nem szokványos, hanem speciális fileok, melyeket az mknod paranccsal lehet létrehozni, de szerencsére a szükségesek már a telepítéskor létrejönnek a telepítő program jóvoltából. Pontosan három és fél ilyen eszközfile-típus létezik. A „fél" abból adódik, hogy a karakteresből kettő is van, de nem tekinthetjük igazán két különbözőnek. Tehát van a blokk eszköz, ezek a háttértárak (merevlemez, floppy, CD-ROM stb). Van az említett két karakteres, melyek annyiban térnek el egymástól, hogy az egyik pufferelt, a másik nem. Az utolsó pedig a FIFO (First In Firts Out) eszköz, mely csővezetékek (pipe-ok) létrehozásához használható, mely igen sokszor használt módszer több program egymás közötti kommunikációjában.

A karakteres eszközök azok, amiken többnyire szövegesen, vagy ahhoz hasonló módon kommunikálunk. Ilyenek a soros portok, digitalizáló táblák, párhuzamos portok, de ilyen eszközön keresztül érhetjük el a valós idejű órát (rtc) is. Linux alatt számos olyan eszköz van,



mely karakteres módszerrel kommunikál. Ide tartoznak a virtuális terminálok is (Virtual terminal). Azért hívjuk virtuálisnak, mert egy fizikai terminálon több is lehet belőlük. Lehetséges, hogy fizikai terminálon egyszerre több virtuális terminált használjunk, azokat egyszerre több dologra használni. Megtehetjük, hogy az egyikben szöveges módban dolgozunk, a másikban a grafikus felület fut, a harmadikban pedig a rendszer naplófile-jait lapozgatjuk. A virtuális terminálok között az ALT funkcióbillentyű vagy CTRL ALT funkcióbillentyű kombinációk lenyomásával válthatunk, ahol a funkcióbillentyűk száma alapján kerülünk a megadott virtuális terminálra. Mozoghatunk előre és hátra az ilyen terminálok között a CTRL ALT nyilak lenyomásával, ahol a „nyilak" a jobbra vagy balra nyilak egyike, attól függően, hogy előre vagy hátra akarunk mozogni. Természetesen az 1. virtuális terminálon megnyomva kétszer a CTRL-ALT-jobbra nyíl gombot egyenértékű az ALT-F3 lenyomásával, vagy ha a futó program ezt kezeli, a CTRL-ALT-F3 lenyomásával.

A virtuális terminálokat a setterm programmal állíthatjuk be kedvünk szerint. Ahhoz, hogy bármi értelmeset csinálhassunk a számítógépünkön, szükség van a virtuális terminálokra, hiszen nélkülük nem kezeli semmi a képernyőt és a billentyűzetet. Azaz csak olyan speciális esetben érdemes kikapcsolni, ahol még arra a csekélyke memóriára is szükség van, melyet ez elfoglal, és biztosan sosem lesz szükség arra, hogy a billentyűzetről kezeljük a gépet. Természetesen hálózatról vagy soros terminálról még ilyenkor is minden további nélkül be lehet lépni.

A Unix típusú rendszerek hagyományosan több terminált használnak, és induláskor automatikusan döntik el, melyik terminál legyen a rendszer konzolja. Immáron a Linux is tudja ezt a funkciót. Alapvetően két lehetőségünk van: vagy a hagyományos billentyűzet–monitor párost használjuk konzolként (azaz valamelyik virtuális terminált), vagy egy soros terminált, mely kábellel kapcsolódik gépünk egyik soros portjához. Hagyományosan ezek vt100 terminálok. Itt most azt kell eldöntenünk, mely terminálok működhessenek konzolként. Azt, hogy ténylegesen melyik, majd a rendszer indításakor kell megadni. Ha bekapcsoljuk a „Support for console on virtual terminal" opciót, akkor a PC-ken megszokott monitor–billentyűzet páros működhet konzolként. Ha a szabványos soros port (Standard/generic serial support) támogatása mellett bekapcsoljuk a „Support for console on serial port" opciót is, a konzol lehet soros terminálon is.

Mint írtuk, a kernel indulásakor dönti el, melyik terminál legyen a konzol. Ezt a következő elv szerint teszi: ha meg lehet virtuális terminálon nyitni a konzolt, az első ilyenen nyitja meg. Ezt teszi legtöbbünk gépe, amikor a kernel elindul. Ha ez nem megy, és lehetőség van soros terminálon megnyitni, megnyitja az első elérhető soros portot mint konzolt, azaz többnyire a ttyS0-t (COM1). Ha ez sem sikerül, nem lesz konzolunk, de a rendszer működni fog. Természetesen a kernelnek átadott paraméterekkel befolyásolhatjuk ezt a döntést. A „console" opció szolgál erre. A paraméterezés: „console=tty1". Itt a tty1 a megnyitandó terminál linuxos neve, jelen esetben az első virtuális terminál (ez az alapértelmezés). Ha „console=ttyS1" opciót adunk meg, akkor a második soros port (COM2) lesz a konzol. Ezeket az opciókat két helyen is megadhatjuk. A legegyszerűbb, ha a LILO promptnál begépeljük a kernel neve után. Tehát ha a „Linux" névvel hivatkozott kernelt (lásd a LILO beállítása) kívánjuk indítani úgy, hogy a konzol a második virtuális terminálon legyen, a megfelelő sor a „LILO boot:" prompt után: „console=tty2".

Felmerül a kérdés, mi van ha soros konzolt szeretnénk, de a LILO a PC-k megszokott monitorára ír, és a billentyűzetről olvas? Itt jön a képbe a második megoldás, a LILO konfigurálása. Ha fix paramétert akarunk megadni (fixen a második soros portra akarjuk tenni a konzolt), egyszerűen a /etc/lilo.conf-ban a megfelelő kernel opcióihoz beírjuk, hogy append „console=ttyS1" (az idézőjelekkel együtt). Ettől kezdve, amíg a LILO promptnál felül nem bíráljuk, a második soros portra kötött terminál lesz a konzol. Ez nem mindig célraveze-

tő, így mi egy sokkal elegánsabb megoldást javaslunk, melynek számos más előnye is van. A LILO ugyanis képes a soros porton kommunikálni, függetlenül attól, hogy a gépünk BIOS-a képes-e átirányítani a kezelést oda vagy sem (vannak olyan PC BIOS-ok, melyek képesek erre). Ha a /etc/lilo.conf általános beállítási részébe felvesszük a „serial=port,bps" parancsot, máris bekapcsoltuk a LILO soros kommunikációját. Egyszerre akár több soros portot is megadhatunk több „serial" opcióval. A „port" a soros port sorszáma nullával kezdve. Azaz a COM1 a 0 sorszámot, a COM2 az 1-es sorszámot kapja. A bps a sebesség bit/s-ban. Ez lehet 110, 150, 300, 600, 1200, 2400, 4800, 9600. Attól függően állítsuk be, hogy mit támogat a terminálunk. Azért nincs szükség 9600-nál magasabb értékre, mert az a pár karakter, amit itt át kell vinni így is elég gyorsan átér. Ez alapján a LILO-ban a „serial=1,9600" opcióval kapcsolhatunk be egy 9600 bps sebességen kommunikáló soros terminált, mely a gép második soros portjára kapcsolódik, és paritás nélkül 8 biten kommunikál. Ha ez utóbbi két adat eltér, azt simán írjuk hozzá a sebességhez. Például ha a paritás „even", és 7 bites a kommunikáció, az opció így alakul: „serial=2,9600E7".

Így máris működik a soros terminálunk, mely lehet egy a hívásokra automatikusan válaszoló modem is, így ha minden kötél szakad távolról is elérhetjük a gépünket, mely éppen újraindult. Ha a rendszer indulása előtt kell elvégeznünk valamit ez nagyon hasznos lehet. Mint tudjuk, ha a LILO nem automatikusan prompttal indul, a billentyűzeten a SHIFT gomb lenyomására kapunk promptot. Ez értelem szerint a soros terminálról nem működhet, így onnan break jelet kell küldenünk ehhez. Érdemes továbbá felemelni a LILO várakozási idejét, hogy a soros terminálról is biztosan időben reagálni tudjunk. Ha a soros portunk a fenti ötlet alapján modemhez kapcsolódik, figyeljünk oda a biztonságra. Általában, ha az így definiált soros konzol nem ugyanolyan biztonságos, mint a hagyományos, védjük a LILO-t jelszóval. Ehhez a „password" opciót alkalmazzuk. Ezt több helyre is írhatjuk: az általános részben minden kernelre vagy LILO bejegyzésre érvényesül feltétel nélkül. Ha az adott bejegyzés (kernel) opcióihoz írjuk, akkor csak ott.

A bejegyzéseknél alkalmazhatunk egy „restricted" opciót is, mely engedi az adott kernelt vagy rendszert bootolni, de opció átadásához már jelszót kér. Mi azt javasoljuk, hogy minden bejegyzés legyen jelszóval védett, és az alapértelmezett kernelhez írjuk be a „restricted" opciót, hogy a rendszerünk jelszó nélkül is elindulhasson. Fontos, hogy ha jelszót adtunk meg a /etc/lilo.conf-ban, akkor rajtunk kívül más ne olvashassa ezt a file-t. Alapesetben mindenki olvashatja, így adjuk ki a „chmod 600 /etc/lilo.conf" parancsot, hogy ezt letiltsuk.

Ezzel végeztünk a LILO megfelelő beállításával. Mostantól akár a hagyományos billentyűzetről, akár egy soros terminálról szeretnénk elérni a konzolt, menni fog. Megvárjuk a LILO promptot, beírjuk a fent leírt kernel opciót, és nézhetjük, ahogy bootol a rendszer.

Térjünk vissza a kernel beállításához. A soros port támogatásra már utaltunk, de azt nem említettük, hogy az is kerülhet modulba. Természetesen ha a soros konzol lehetőségére is szükségünk van, a kernelbe kell kerülnie, de minden más esetben modulba tegyük. A következő pont az extra soros port opciók beállítása, ha szükségünk van erre. A kernel képes négynél több soros portot kezelni. Ha több van, kapcsoljuk be az „Extended dumb serial driver options" mellé a „Support more than 4 serial ports" opciót is. A mai PC-k többségében kevés az elérhető megszakítások száma (az Intel Xeon processzoros gépekben ez már 64, ami kész felüdülés az évtizedes hagyományokkal rendelkező 16 megszakításhoz képest). Ez akkor probléma, ha több megszakítást igénylő eszközt szeretnénk betenni a gépbe, mint ahány szabad megszakítás még van. A soros port pedig egy ilyen megszakíts igényes eszköz. Hogy spórolhassunk, lehetőségünk van több soros portot egy megszakításon használni. Ehhez kapcsoljuk be a „Support for sharing serial interrupts" opciót.

Javasoljuk, hogy olyan portokat tegyünk egy megszakításra, melyeken vagy kicsi a forgalom, vagy ritkán forgalmaznak egyszerre. Például nyugodtan kerülhet egy megszakításra a szünetmentes tápunkkal kommunikáló port mással, de nem javasoljuk, hogy két modem kerüljön egy megszakításra. Ha módunkban áll, ne használjunk egy megszakítást több porthoz. A következő lehetőség, hogy a kernelre bízzuk a soros portok megszakításának megtalálását. Ezt mindenképpen kézzel kell megadni a BIOS-ban vagy a soros kártya jumpereivel, de a Linuxnak is tudnia kell róla valahogy. Szokás szerint itt is több lehetőség közül választhatunk. Ha a soros port megszakítása a szabványos helyen van (COM1/IRQ4, COM2/IRQ3), semmi dolgunk, ezt veszi a kernel is alapértelmezésnek. Ha ettől eltér, például beteszünk egy harmadik soros portot, ami a COM3 lesz, de az IRQ4 helyett (amit a COM1 már használ) a még szabad IRQ5-re tesszük a megszakítást, a kernel nem fog tudni róla, és a port nem fog működni.

A biztonságos és a 2.0-ás kerneleknél megszokott megoldás, hogy a setserial paranccsal megadjuk a kernelnek a szükséges paramétereket. Nem annyira megbízható (okozhat lefagyást a boot során), ha itt bekapcsoljuk az „Autodetect IRQ on standard ports" opciót,

ennek hatására amikor a rendszer bootol, vagy betöltjük a soros port modulját, az automatikusan megpróbálja megtalálni a megfelelő port-megszakítás párost. A mi tapasztalatunk az, hogy a legtöbb esetben ez működik, de használatát csak akkor javasoljuk, ha előtte alaposan teszteltük azt. Komoly szervereken, vagy ahol amúgy sem változik gyakran a soros portok beállítása inkább a setserial használatát javasoljuk.

A további opciók a hagyományos soros portokhoz lehetővé teszik többportos soros csatlakozó eszközök használatát (Support special multiport boards, Support the Bell Technologies HUB6 card). Ezután a nem szabványos soros port vezérlők támogatása következik. Ide tartoznak azon egyszerre sok (8–16–32 stb.) soros port kiszolgálására képes eszközök, melyeket behívószervereken vagy sok terminált kiszolgáló gépeken szoktak alkalmazni. Továbbá itt találunk drivert egyes speciális, például nagysebességű soros portokhoz is. Mindenkinek magának kell megtalálnia, hogy egy-egy eszközhöz melyik driver is kell.

Természetesen itt is a modularitást javasoljuk, és a már leírt próbálkozást a modulokkal.

A következő érdekesség a „Unix98 PTY" támogatás. Ez a szabványos Unix PTY, azaz pszeudo terminálok kezelését szolgálja, melyhez felhasználói szintű támogatást csak a 2.1-es GNU Libcben találunk, azaz komoly munkára használt gépen még nem. Célja, hogy megvalósítsa (illetve lényegileg ez már megvalósult) a Unixokon megszokott master-slave pszeudo terminálrendszert. Sokkal fontosabb a párhuzamos portra csatlakoztatható nyomtatók támogatása, mely létfontosságú a nyomtatáshoz. Figyelmes olvasóinknak feltűnhetett, hogy immár a harmadik párhuzamosport-kezelő kernelrész kerül elő, a párhuzamos portra csatlakoztatható lemezegységekről nem is beszélve. A 2.2-es kernelben részeire szedték és egységesítették a párhuzamosport-kezelést.

Régen (a 2.0-ás kernelben) az lp driver – mely a nyomtatásért volt felelős – kezelte a hardvert és a nyomtatót egyaránt. Ez így erősen hardverfüggő volt, és mivel a Linux nemcsak Intel kompatibilis processzorokon, hanem nagyobb, 64 bites architektúrákon (SPARC, ALPHA, PPC stb.) is fut, egységesíteni kellett ezt is. Ezek szerint most találunk egy alacsony szintű párhuzamos port kezelőt, mely hardverfüggetlenül ad egy szabványos párhuzamosport felületet. Ez a driver inicializálásakor betölti az adott hardvernek megfelelő kezelőprogramot. Ezt a felületet használja ezután minden párhuzamos porton kommunikáló program, így a nyomtatókezelő is. Ha minden párhuzamos porttal kapcsolatos funkciót modulba tettünk (amit érdemes tenni), és egy nyomtatás közben megnézzük az aktív modulok listáját (lsmod), a következőket találjuk benne, ha PC-n dolgozunk: parport, parport_pc, lp. Pontosabban ezek tartoznak ide, ezek az előzőekben leírt kezelőprogramok. Ha bekapcsoljuk a „Support IEEE1284 status readback" opciót, a nyomtatókezelő programunk, ha a nyomtató támogatja az IEEE1284 állapotlekérdezést, a naplóba rendszeresen beírja a nyomtató állapotait, és minden olyan adatot, melyet az visszaad.

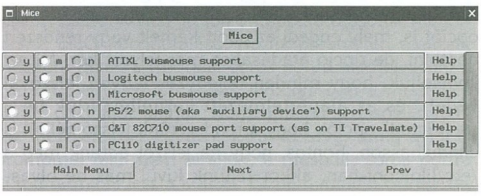
A nyomtatórész után az egerek következnek. Természetesen nem a soros portra csatlakozó egerek, azokhoz nem kell külön kerneltámogatás. Itt a speciális portra, mint például a PS/2 port csatlakozó egerekről van szó. Ha itt bekapcsoljuk, az egerek menüben beállíthatjuk, milyen egeret támogasson a kernel.

Találunk még pár karakter alapú eszközt, melyhez itt kell bekapcsolni a támogatást. Ide tartozik a „QIC-02" szalagos egység, mely egy egyedi protokollal rendelkező DAT, a watchdog, mely a rendszer állapotát hivatott figyelni, és ha az nem válaszol, rebootolja azt. Ha ezt bekapcsoljuk, két menüvel később találkozni fogunk az egyedi eszközök kezelőjével.

Ha szükséges, Linux alól elérhetjük a PC-nk nem felejtő memóriáját, vagyis a CMOS-t. Ennek segítségével matathatunk programból a BIOS adataiban, lekérhetjük vagy módosíthatjuk a CMOS időt. Természetesen ehhez külön programot kell írni, ami a /dev/nvram eszközt használja. Modulba fordítása nem jelent problémát, így ha szükséges elérhetjük ezt a funkciót. A CMOS memória matatását nem javasoljuk, mivel ezzel alaposan elronthatjuk a BIOS beállítását. Hasonló funkciója van a valósidejű óra támogatásnak (Enhanced Real Time Clock Support), mely az alaplap óráját hivatott elérni. Ha bekapcsoljuk, a /proc/rtc alatt olvashatjuk ki az RTC adatait, és a /dev/rtc eszközzel vezérelhetjük, ha van hozzá megfelelő programunk.

Egerészés

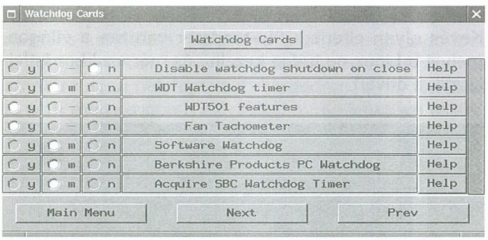
Amint a karakteres eszközök beállításánál azt már említettük, bizonyos speciális protokollt „beszélő" egerekhez külön támogatás kell a kernelbe.



Ide tartoznak a PS/2-es egerek, Logitech és Microsoft bus-egerek. Persze ez utóbbi két gyártó újabb egerei már vagy soros vagy PS/2-es porton kapcsolódnak a géphez, de pár régebbi típushoz kellhet ez a driver. Ellentétben a 2.0-ás kernelekkel, a PS/2 egér-port támogatást már nem lehet modulba fordítani. Teljességgel felesleges is ezt tenni, mivel a PS/2-es portra csak a gép bekapcsolását megelőzően szabad eszközt dugni vagy lehúzni, így vagy van ilyen egerünk, vagy nincs, ez menet közben nem változhat. Nem is javasoljuk az ezzel való kísérletezgetést, mert rövid úton az alaplap tönkremeneteléhez vezethet.

A házőrző kutya

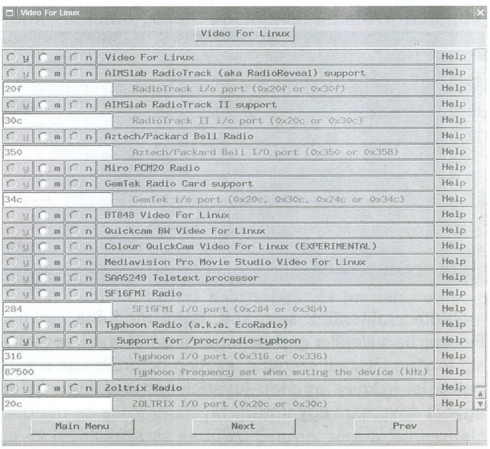
Mint ezt is említettük, használhatunk watchdog eszközt, mely „figyeli" a gépünket, és ha kell újraindítja. Ez a figyelem annyit jelent, hogy ha a gép nem felel, vagy a terhelés egy bizonyos előre meghatározott szint fölé nő, újraindítja a gépet. A hardver watchdogok ezen felül más funkciókat is elláthatnak, típusa válogatja.



Találunk néhány kezelőt hardver watchdogokhoz, és egy szoftveres watchdogot is. Természetesen csak a hardveres az igazán biztos, hiszen az ha kell fizikailag indítja újra a gépet, míg a szoftveres esetén a Kernel próbál resetelni, ha a felhasználószintű programból nem kapja meg a megfelelő információkat. A szoftver watchdog használatához tegyük fel a watchdog csomagot a Debian CD-ről, és kapcsoljuk be a „Software Watchdog" opciót.

Videózzunk!

Egy kis szórakoztatás a sok komoly dolog után, bár biztos van akinek ez is a munkát jelenti. A 2.2-es kernelben jelent meg először, de már külső csomagként a 2.0-ához is elérhető volt a „Video For Linux" rendszer, azaz a videodigitalizáló, rádió- és tévétuner kártyák támogatása. Külön ki kell emelnünk a Bt848-as chipsetet, melyhez a támogatás már igen régen megvan és talán az egyik legjobban működő videograbber. A minőségét olyan név fémjelzi, mint a Miro, melynek PC/TV kártyájában is ez a chipset található. Állíthatjuk, hogy szép és jó képet ad a képernyőre. Hasonlóan kellemes élmény, bár eggyel gyengébb minőség (no nem a gyártó miatt, hanem a technológia más) a Quickcam, melyhez szintén jó támogatást találunk itt. Ezzel könnyedén készíthetünk internetes videokonferencia rendszert. A kernel támogatja mind a fekete–fehér, mind a színes változatot. Ezek mellett még említést érdemel a Mediavision Pro Movie Studio is, de találunk teletext kártya és sokféle rádió tuner támogatást is.

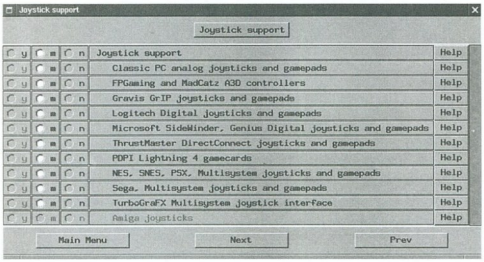


Természetesen a videograbberhez, de a többi funkcióhoz is szükséges a felhasználói-szintű kezelőprogram. Az egyik legjobb a létezők közül az a tévéprogram, mely a Bt848-as chipsettel rendelkező kártyákhoz készült. Kétféle változatban létezik: az xawtv a X11 grafikus felületen működik és még az fbtv a framebuffert használja. Ez utóbbiról, mármint a framebufferről még lesz szó. Itt csak annyit, hogy ez egy konzol driver, mely képes grafikai megjelenítésre minden VESA2-es kártyán. Mindkét változat képes a teljes-képernyős megjelenítésre, azaz kényelmesen tévézhetünk a monitorunkon.

Joystick támogatás

Nem állíthatjuk, hogy a Linux egy tipikus játék-platform, de az alapjai már megvannak, hogy azzá válhasson. Egyrészt a teljesítőképessége, mely a játékoknál

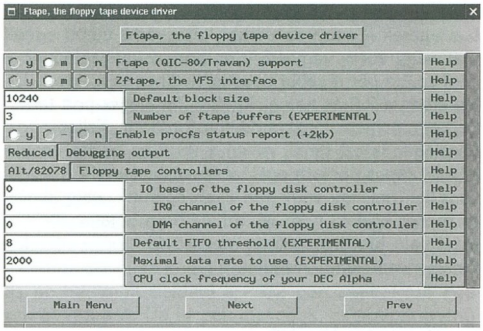
igen fontos, másrészt az ugyan még fejlesztés alatt álló, de egy gyors és szabványos grafikai megjelenítést lehetővé tevő platformfüggetlen rendszer a GGI is a fejlesztők rendelkezésére áll. Mindezt „megspékeltetjük" az OpenGL támogatással, és máris nyomulhatnak az akció- és látványorientált játékok. És ha lesz sok játék, kell sok joystick. A Commodore 64-en felnevelkedett nemzedék számára elsőre ugyan furcsa volt, hogy ahány joystick, annyiféle kezelőszoftver. Mit is várhatnánk a PC világtól... Mindenesetre a Linux sem akar lemaradni, és mivel akad már néhány játék, és játékőrültek a linuxos társadalomban is vannak, joystick támogatás is van jó néhány botkormányhoz.



Szinte említenünk sem kell, hogy modulba tegyük. Persze, ha valaki a gép bekapcsolásától a kikapcsolásig folyton játszani szeretne, esetleg befordíthatja a kernelbe is. Igaz ez az alap-joystick támogatásra és az egyedi driverekre egyaránt.

Floppy alapú szalagos egységek

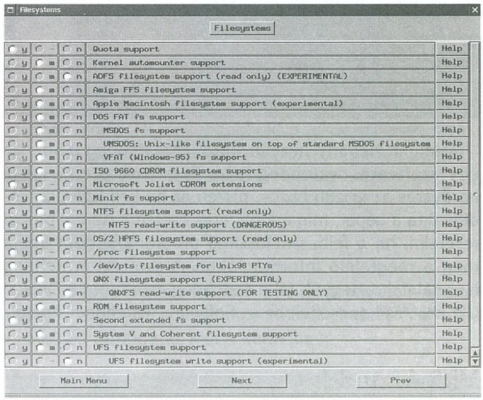
Az intelligensebb SCSI eszközök helyett nem csak a CD-ROM-ok és a merevlemezek gyártói találtak ki olcsóbb, de butább eszközöket, megtörtént ugyanez a szalagos egységekkel is. Tény, hogy ezek ára sokkal emberbarátibb, mint a SCSI verzióké, de cserébe több erőforrást vonnak el a géptől (főleg a processzortól), és speciális kezelőprogramot igényelnek a kernel szintjén. Ezek egytől-egyig a hagyományos floppy kontrollerre épülnek. Van amelyiket egyszerűen a floppy csatolóra kell kötni, de vannak (például a Seagate Tape Store 3200, Iomega Ditto 3200, Exabyte Eagle TR-3), amikhez saját vezérlőt is adnak. Ha az „Ftape, the floppy tape device driver" menüben modulba tesszük az „Ftape" opciót, valamint a „ZFtape" opciót, máris használhatjuk az eszközünket. Találunk még jó néhány beállítást, de ezek mikéntjét megfelelő tapasztalat hiányában itt most nem tudjuk ismertetni. Ami az igazat illeti, mi a SCSI DAT hívei vagyunk.



Fontos, hogy ha PCI-buszos rendszerben szeretnénk használni ezeket, olvassuk el a drivers/char/ftape/README.PCI leírást a kernel forráskódban.

File-rendszerek

Kevés olyan elterjedt file-rendszer van ma a világon, melyet a Linux ne tudna kezelni, és már azokhoz is fejlesztik a drivert.



A „Filesystem" menüben megtaláljuk a hálózati file-rendszerek kivételével az összeset. A hálózati file-rendszerekhez egy külön menü van. Ha sorba vesszük a lehetőségeket, először a „Quota" támogatást láthatjuk. Ez nem igazán egy file-rendszer, hanem a file-

rendszerekhez egy extra kiegészítő: felhasználóra és csoportra egyedileg szabályozhatjuk, melyik partíción mennyi helyet foglalhat el. Mindezt elég jól, sok beállítási lehetőséget hagyva. Mivel ennek kezelése kernelszinten zajlik, a felhasználónak reménye sem lehet arra, hogy megkerülje, és ez jó (legalábbis a rendszergazda szemszögéből).

Felhasználónként beállíthatjuk, hogy egy-egy partíción hány könyvtára, hány file-ja lehet, vagy mennyi helyet foglalhat el byte-ban megadva. Természetesen semmit sem kell megadni, nem kell limitálni. Mi például nem találkoztunk még olyan beállítással, ahol a könyvtárak számát limitálták volna. Bár, ha a példa kedvéért a /home-ot tartalmazó partíción valakinek maximum egy könyvtárat engedélyezünk, könnyedén megtiltottuk neki, hogy alkönyvtárakat hozzon létre. Ez a rendszer tarkítva van azzal, hogy két korlátozó adatot adhatunk meg: egy quotát és egy limitet, valamint egy ezekhez tartozó türelmi időt. A quota egy olyan érték, melyet túlléphet a felhasználó. A limitet nem lépheti túl. Ha a limit nagyobb, mint a quota, a rendszer a következőképpen fog viselkedni: ha a felhasználó túllépi a quotáját, a türelmi időben meghatározott napig szabadon dolgozhat, mindössze a limit mértéke korlátozza. Ha a türelmi időt meghaladó ideig folyamatosan túllépi a quotát, a rendszer letiltja számára az új helyfoglalást mindaddig, amíg törléssel le nem megy a quota alá. Ezzel máris lehetővé tettük, hogy ideiglenesen viszonylag sok helyet használhassanak a felhasználóink, de hosszútávon ne tárolhassanak sok adatot a gépen.

Ugyanez működik csoportra is. Mint tudjuk, Unixokon, így Linuxon is minden file-nak van egy tulajdonosa és egy csoportja. Csoportquota esetén értelemszerűen az egy csoport tulajdonában lévő file-ok a mérvadók. Ezzel, ha a felhasználóknak közös csoportjuk van, a csoport helyfoglalását korlátozhatjuk. Mindezt keverhetjük is, azaz lehet felhasználói és csoportquota egyszerre. Ahhoz, hogy a quota működjön, egyrészt itt be kell kapcsolni, másrészt a csatolási opciókhoz (/etc/fstab) hozzá kell adni felhasználóiquotához az „usrquota", csoportquotához a „grpquota" opciókat. A quoták kezelését a quota csomagban található edquota, quota, quotacheck, quotaon és quotaoff programok végzik. A quotacheck ellenőrzi naponta a quotákat (mármint Debian alatt fut naponta, mert a cron elindítja), a quotaon bekapcsolja, a quotaoff kikapcsolja a quotákat egy adott file-rendszeren. Feleslegesen ne kapcsoljuk be a quotát, mivel valamennyire lassítja a file-rendszer kezelését. A quota ki- és bekapcsolását a Debian induláskor és leálláskor automatikusan elvégzi, ha a megfelelő opciókat beírtuk az fstab-ba. A felhasználók quotáját az edquota paranccsal módosítani, a quota paranccsal pedig megnézni tudjuk. Ez utóbbit a felhasználó maga is használhatja a saját quotájának ellenőrzésére.

Továbbra sem igazi file-rendszer szintű, de szokás szerint igen hasznos funkció a „Kernel automounter support". Mint az tudvalevő, Unixok alatt az összes file-rendszer a root (/) file-rendszer alatt van valahol csatolva, és ahhoz, hogy lásuk a file-okat valahova csatolni kell a háttértárat. De az is igaz, hogy nem túl egészséges a felhasználóknak korlátlan jogokat adni a csatolások megváltoztatásához. Viszont sokszor van szükség mobil eszközökre, mint a floppy vagy a CD-ROM. Ezekhez muszáj valamiféle hozzáférést biztosítani. Az fstab „user" opciójával lehetővé tehetjük, hogy a felhasználó a „mount" paranccsal csatoljon, az „umount" paranccsal leválasszon eszközöket, mint például a CD-ROM-ot. Ez működik, de nem túl kényelmes. Ráadásul a CD-t nem tudjuk kivenni, amíg az csatolva van. Ez logikus, de ha kellően messze van a terminálunk a szervertől, amiben a CD van, nem fogjuk díjazni, ha vissza kell sétálni umountolni, majd odamenni kivenni. Szerencsére megtalálták a megoldást, és ezt hívjuk „autofs"-nek, azaz automatikus file-rendszernek. A felhasználói-szintű automount program és ez az opció együttesen megoldják a problémát. Például, ha ezt használjuk, és jól van beállítva a CD-ROM-egységünk, a /amnt/cd könyvtárba belépve máris látjuk a CD-t, és amint kiléptünk, máris ki tudjuk azt venni.

Ez valójában úgy történik, hogy a rendszer, pontosabban az automount program automatikusan csatolja és leválasztja az eszközt. Szerencsére a felhasználó ebből csak annyit vesz észre, hogy ha kell látja a CD-t, ha nem akkor kiveheti. Ha megnézzük a /amnt könyvtár tartalmát, feltűnhet, hogy teljesen üres. Ez így rendjén is van, de próbáljunk mégis belépni a CD alkönyvtárába. Ekkor (ha be van konfigurálva és van CD a meghajtóban) a könyvtár létrejön és benn is vagyunk. Ha kilépünk (és senki más nem használja már), akkor a leválasztás után a könyvtár is eltűnik. Ugyanilyen módon kezelhető a floppyegység is a /amnt/floppy hivatkozással. Az automount programot kedvünkre konfigurálhatjuk a /etc/auto.master és alapesetben a /etc/auto.amnt file-okban.

A továbbiakban már valódi file-rendszerek következnek. Amint a modulokról szóló részben már említettük, ahhoz, hogy a rendszer elindulhasson, a natív linuxos file-rendszert be kell fordítani a kernelbe. Ez esetünkben az ext2, azaz „Second extended filesystem". Továbbra is igaz, hogy NFS-gyökérrel induló diskless gép kernelébe felesleges, ez is mehet modul-

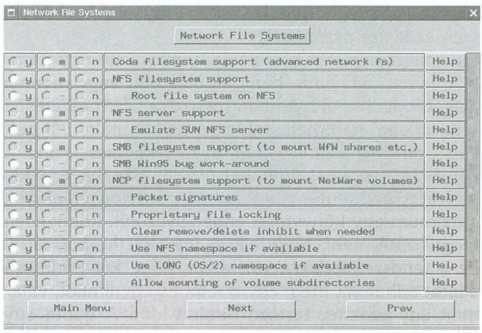
ba. Általában azonban kell. Találunk még egy „Extended filesystem"-et is, ez egy régebbi, már nem használt file-rendszer. A Minix egy Unixszerű rendszer filesysteme, a Linux hagyományosan használja, főleg ha kis helyen kell elférni. Jó tudni, hogy Linus Torvalds, a Linux „apja" Minixen kezdett tanulni, és annak hiányosságai által inspirálva (no meg mert kíváncsi volt az Intel processzorok programozására) kezdett bele a Linux fejlesztésébe.

Amire még szükségünk lehet az Amiga, Macintosh, NTFS, HPFS, QNX és UFS file-rendszerek lenyűgöző tömegében, az a DOS FAT, VFAT és FAT32 támogatás, melyekhez találunk egy UMSDOS kiegészítést, mely képes a Unix jogosultságok tárolására és használatára ezen file-rendszereken. Semmi ördöngösség nincs benne, egy sima file-ban tárolja könyvtáranként ezen információkat, melyek Linux alatt az umsdos kiegészítő használata közben nem látszanak, de amúgy igen. Egyszóval ez nem véd meg minket semmitől, ha DOS-t indítunk. A CD lemezek olvasásához az ISO 9660-as file-rendszerre lesz szükségünk, melyhez a beépített RockRidge kiegészítés adja hozzá a hosszúfile-név és a unixos jogosultság támogatást, valamint bekapcsolhatjuk a Microsoft Joliet kiegészítést. Arra még keresik a szakemberek a választ, hogy a Microsoftnak miért nem felet meg a RockRidge...

Egy speciális file-rendszer a /proc file-rendszer. Erről, legalábbis a könyvtárról és az alatta levő pár dologról már volt szó. Ha itt nem kapcsoljuk be a támogatást hozzá, sok problémánk lesz a rendszerben, így tegyük ezt bele. Szintén volt szó a Unix98 PTY támogatásról, melyet ma még csak a fejlesztői Linuxokon használnak, így erre legfeljebb a kísérletezés erejéig lehet szükségünk. Szintén specifikus dolog a ROM file-rendszer, mely elsődlegesen initrd készítéséhez hasznos.

A hálózati file-rendszerek

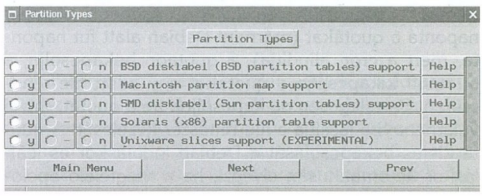
A Linux képes számos file-szerverhez kapcsolódni. Ahhoz, hogy a szerverek által közzétett file-rendszereket elérhessük, szükség van a kernelben ezek támogatására. A „Network File Systems" menüben kapcsolhatjuk ezeket be, illetve állíthatjuk be a kívánt opciókat. A legismertebb ilyen talán az NFS (Network File System), a Unixok hagyományos hálózati file-rendszere. Nagy hibatűrésű file-rendszer, mely akkor is képes fennmaradni, ha a szervert órákig nem látja a kliens. Sajnos ilyenkor a kliensen megáll az élet, de nem is leválasztott működésre találták ki az NFS-t. Arra a CODA való, mely egy elosztott file-rendszer, és számos funkciója között ez is szerepel. Új fejlesztés, beállítása nem könnyű feladat, de számos olyan szolgáltatása van, melyet eddig mással nem lehetett



megvalósítani, vagy legalábbis nem ilyen hatékonyan. Használatához a szerveren a coda-server, a kliensen a coda-client csomagot kell telepíteni. Visszatérve az NFS-re, ha azt a kernelbe fordítjuk, aktivizálódik a „Root file system on NFS" opció. Ha azt is bekapcsoljuk, az így elkészített kernel alkalmas lesz (feltéve ha az eddigiekben ebben a témában leírtakat elvégeztük) a hálózatról történő bootolásra. Ha ez nem kell, az NDS-t is modulba tegyük.

Az NFS-szerver támogatás új a 2.2-es Kernelben. Eddig felhasználószintű programok végezték az NFS-szolgáltatást, most már lehetőségünk van ezt a kernelre bízni. Ehhez a knfs csomag kell, ami képes az új NFS-szerver funkciók kezelésére. Ha az „Emulate SUN NFS server" opciót is bekapcsoljuk, exportálni tudunk olyan részt is a file-rendszerből, mely már elve máshonan lett csatolva.

A PC-k világában legismertebb két hálózati file-rendszer az SMB (mai divatosabb nevén CIFS, azaz Common Internet File System) és az NCP. Az előbbi a Microsoft „találmánya", és Windows gépek TCP/IP-s könyvtár- és nyomtatómegosztására szolgál (Netbios over TCP/IP). Talán már mindenki hallott a Samba-



szerverről, mely ennek a világon egyik legjobb, és természetesen szabad szoftver implementációja (megtalálható a CD-n). Az utóbbi, azaz az NCP a Novell Netware kötetek csatolását szolgálja, és sok újdonsággal rendelkezik a 2.2-es kernelekben. Ezek főleg a menüben látható opciók, melyek igény szerint állítandók be. Az NFS és OS/2 hosszúfile-név kiegészítés talán az, melyet mindenki be akar majd kapcsolni.

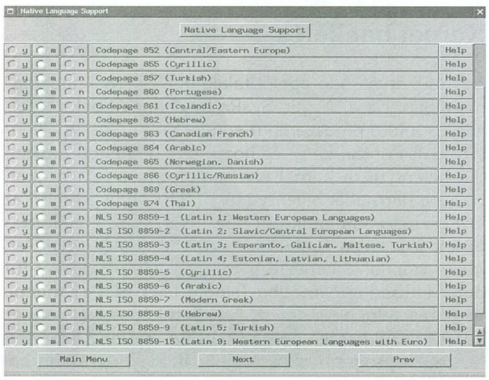
Viszonylag szorosan kapcsolódik a file-rendszerekhez a támogatott partíció-típusok beállítása. Talán kevesen tudják, de a PC-ken használatos particionálás nem egy általános módszer, sőt, a PC-ken sem mindenhol használják. Vannak ennél sokkal fejlettebb rendszerek is, de a Linux mégis alapvetően ezt használja. Ez talán annak köszönhető, hogy a DOS annyira elterjedt a 90-es évek elején. Az viszont biztos, hogy mára már a Linux is kezel számos más partíciós táblát, köztük a BSD disklabelt, a Macintosh partíciós tereket, a Sun partíciós táblákat, a Solaris partíciós tábláját és a Unixware által használt úgynevezett „Slices"-t. Ezeket modulba fordítani értelmetlen. A DOS-os partíciós tábla kezelése olyan szinten van integrálva a kernelbe, hogy kikapcsolni sem lehet. A többit akkor kapcsoljuk csak be, ha szükségünk lehet rá, ellenkező esetben feleslegesen foglalja csak a memóriát.

Nyelvtámogatás

A világ különböző nyelveihez különböző betűk tartoznak. Mi még a szerencsésebbek közé tartozunk, mert ékezetes betűink jó része megtalálható egy eredeti PC-s kódtáblában is, de a cirill, japán vagy kínai jeleket használóknak már nehezebb a dolguk. Tény, hogy a számítástechnika nyelve az angol, és ez nagyon erősen rányomta bélyegét a rendszerekre.

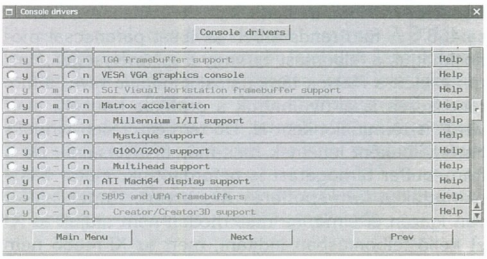
Szerencsére a Linux ma már a legkülönfélébb szabványos nyelveket támogatja.

A PC-s kódlapoktól az ISO szabványokig mind megtalálható a kernelben. Azért itt van, hogy a speciális karaktereket tartalmazó file-okat korrektül tudja kezelni. Mi azt javasoljuk, hogy az összes kódlapot tegye modulba, így bármikor bármelyiket tudja használni, ha szükségessé válik. Ezek kezelése teljesen automatikus.



Konzolkezelés

A terminálóknál már írtunk a konzolról. Itt állíthatjuk be, milyen konzolkezelést szeretnénk.



Ez itt csak a gépre fizikailag kötött monitorra vonatkozik, a soros terminál, mint konzol nem a Linux kernel hatásköre.

Alapvetően, PC-n kétféle konzolt használhatunk: a hagyományos szöveges konzolt (VGA text console), vagy a VESA grafikus konzolt. Ez utóbbival működik a frame buffer support (Support for frame buffer devices), mely olyan hardver platformokról lett a PC-s változatba átörökítve, melyeknél nem létezett a szöveges változat (Sparc, PowerPC gépek). Nagy előnye, hogy szabvány VESA2 módban használva kellemesen nagy felbontásokat lehet vele elérni minden irányban. Emellett még egy egységes grafikus programozói felületet is ad, így végre megszabadulhatunk az SVGAlib ártalmaitól. Készült már framebuffert használó X-szerver is, de a már említett fbtv tévétuner program is ezt használja a megjelenítéshez. Akár mindkét támogatást (VGA text, framebuffer) beleforditathatjuk a kernelbe, és boot előtt eldönthetjük, melyiket választjuk. Legalább az egyiknek benne kell lennie a kernelben, hogy a konzolunk működjön.

A LILO promptnál vagy a lilo.conf-ban a „vga" opcióval állíthatjuk be a kívánt módot. A „vga=ask" hatására a kernel megkérdezi, mit szeretnénk, és a választáshoz egy menüt is ad. Kódok formájában kell megadni a kívánt értéket, mely kódok a kernel dokumentációjában megtalálhatók.

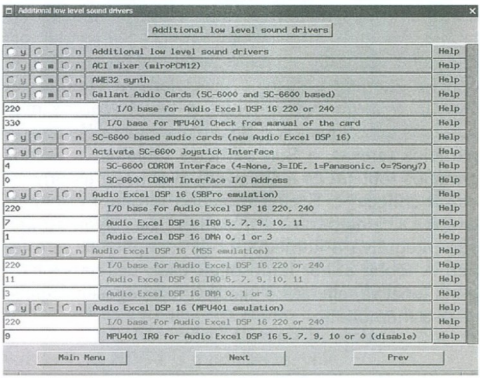
Számos inaktív (csak más hardver platformon értelmes) opció után a „VESA VGA graphics console" opció következik. Ez egy teljesen grafikus konzol, mely szabvány VESA2 módok használatával minden ezt ismerő videokártyával működik. Találunk hozzá gyorsító funkciókat, amit néhány jobb minőségű videokártyához használhatunk, mint például a Matrox, vagy az ATI Mach64 kártyák (legalábbis PC-n).

Ha a VESA grafikus konzolt használjuk, és ezt megadjuk a kernelnek („video" opció), akkor egy kedves pingvin képe kísér minket a boot során amellett, hogy a megadott felbontásban indul el a rendszer. Ha Matrox kártyát használunk, az alábbi opció 1152x768, truecolor módban indítja a kernelt: „video=matrox:vesa:403". A futó rendszerben az fbset paranccsal módosíthatjuk a felbontást és/vagy a színmélységet. Matrox esetén a Matrox támogatást modulba téve, a futó rendszeren is aktiválhatjuk a grafikus konzolt, akár virtuális terminálonként is. De természetesen virtuális terminálonként lehetnek más-más beállításaink. Ha a framebuffer támogatást is befordítottuk, akkor használhatjuk a farmebufferes X-szervert. Ennek akkor van igazán értelme, ha más módon nem tudjuk az X-et kellő felbontásban indítani, mert az XFree86 nem támogatja a kártyánkat, de a kártya ismeri a VESA2-t. Egyébként nem, mert az XFree86 natív driverei sokkal gyorsabbak és stabilabbak, mint a még fejlesztői stádiumban levő framebuffer.

Hangkártyák

A hangkártyatámogatás mindig egy fájó pont volt a Linuxban. Mára ez szerencsére megváltozott. Valamelyest javult a kerneltámogatás a 2.0-hoz képest, de még mindig nem az igazi. Aki komolyabban foglalkozik a témával hamar rátalál az ALSA (Advanced Linux Sound Architecture) driverre, mely sokkal jobb, többet tud és kényelmesebben használható. Ennek ellenére, ha valaki a kernelben lévő drivert szeretné használni, megvan rá a lehetősége. Ha modulba teszi, arra is, hogy hol ezt, hol azt használja, és miért ne tennénk modulba. Ki látott már hangkártyáról bootoló számítógépet?!

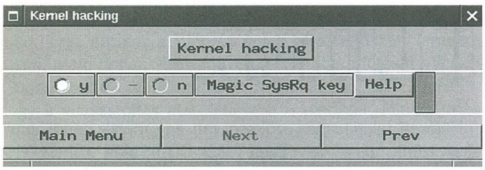
Mi is az ALSA használatát javasoljuk, és természetesen a CD-n megtalálható az ALSA a telepítő kernelhez fordítva és forráskódban is.



Csak a teljesség kedvéért leírjuk, milyen ismertebb hangkártyát támogat a kernel. Az ALSA ezeket is, és még sok mást is támogat. Tehát a lista: Ensoniq, SoundBlaster, Turtle Beach MultiSound Pinnacle, Gravis, Crystal, Yamaha, Aztec Sound Galaxy stb.

Kernel hackelés

Egyedül árválkodó opciót találunk a „Kernel hacking" menüben. Ha a „Magic SysRq key" opciót bekapcsoljuk, és a futó rendszeren megnyomjuk a SysRq billentyűt (Alt-PrintScreen), és hozzá egy megadott harmadik billentyűt, a kernel elvégez bizonyos feladatokat. Ha ez a harmadik billentyű az ENTER, egy listát ad az elvégezhető feladatokról (igen rövid, csak emlékeztetőnek jó). Ezen funkciók között találjuk a sync-et, mely azonnal a lemezre ír minden adatot, de így tudunk karbantartó módba (single user mode) menni, vagy resetelni a rendszert. Emellett képes leállítani az adott konzol összes programját, ha épp egy ilyen felhasználói program okozott galibát. Egyszóval, ha baj



van, segíthet, érdemes bekapcsolni, és elolvasni a Documentation/sysrq.txt file-t a kernel forráskódban, hogy mit és hogyan lehet ezzel tenni.

Fordítás és telepítés

Ezzel a kernel beállítások végére értünk, és ha az Olvasónknak még van türelme, leírjuk, hogyan fordítsuk le és telepítsük fel Debian alatt a kernelt. Persze mielőtt a fordításhoz jutnánk, lépjünk ki a beállító programból úgy, hogy az mentse el a beállításokat. Esetleg érdemes egy alternatív file-ba is lementeni, későbbi használatra.

Ha ez megvan, és még emlékszünk, hogy a /usr/src/kernel-source-2.2.4 könyvtárban állunk, adjuk ki a következő parancsot:

„make-kpkg --zlmage --revision=sajat.1 kernel_image"

Lássuk mit is jelent ez. A make-kpkg a kernel csomagot készítő program neve. Ennek a telepítését javasoltuk „néhány" oldallal ezelőtt... Ezután az opciók jönnek, kivétel nélkül két kötőjellel bevezetve. A --zlmage jelenti, hogy sima zlmage file-t szeretnénk kapni bízva abban, hogy a kernel 512k-nál kisebb lesz.

Ha ez nem jött be, akkor nem elég moduláris kernelt csináltunk, érdemes ismét belenézni a beállításokba. Ha ragaszkodunk a beállításainkhoz, akkor a --bzlmage, vagy az opció elhagyása megoldja a problémát. A --revision opcióval adunk egy egyedi nevet a kernelünknek. Erre azért van szükség, mert így egy esetleges később disztribúció frissítés gyári kernele nem fogja felülírni a miénket.

Használjuk a név.sorszám formát az átláthatóság kedvéért. A sajat.1 helyett természetesen írhatunk bármit. Arra figyeljünk, hogy legközelebb is ez marad, mindaddig, míg a debian könyvtárat le nem töröljük, de azt nyugodtan megtehetjük (rm -rf debian), mivel újra fogja generálni. A kernel_image jelenti, hogy egy bináris kernel csomagot szeretnénk kapni. A kernel_headers kernel header csomagot, a kernel_source kernel forráskód-csomagot generál. Ha a fordítás sikeres, az új csomag kernel-image-2.2.4_sajat.1_i386.deb néven az egyel felettünk levő könyvtárban, azaz esetünkben a /usr/src alatt lesz fellelhető. Ezt a „dpkg -i kernel-image-2.2.4_sajat.1_i386.deb" paranccsal telepíthetjük. Ha ez megvolt, adjuk ki a lilo parancsot, és jöhet a reboot (shutdown -r now).

Ha szerencsénk van, és mindent jól sikerült beállítani, a rendszerünk elindul az új kernellel. ■

Slapic
slapic@vogel.hu

Mobil Linux

Egyre többek számára válik munkájukban pótolhatatlan társsá Intel processzorra épülő noteszgépük. A méret és a hordozhatóság azonban sok esetben kompatibilitási problémákhoz vezet, mivel a noteszgépek világában számos olyan (PCMCIA) vezérlő használatos, ami szokatlan az asztali PC-knél.

Ezek valamennyi operációs rendszer esetében problémák forrását jelentik. Nem véletlen, hogy az egyik legelterjedtebb PC-s operációs rendszernek is sok esetben a gyártó által saját hardveréhez hangolt speciális noteszgép-verziója kerül telepítésre. A szabadszoftverek világában, az, hogy elérhető a forráskód, lehetővé teszi, hogy ezeket a problémákat könnyebben áthidaljuk. Így ma már számos noteszgépen igen jól fut a Linux. Nekünk egy Portocom noteszgépet sikerült kipróbálnunk, és az itt leírtak egy része az azon gyűjtött tapasztalaton alapul. Természetesen a legtöbb megoldás általános, de sok egyedi vonás létezik, melyet csak az adott gyártó gépét kipróbálva célszerű ismertetni.

Telepítés

Egyre több noteszgépben bukkan fel CD-ROM-meghajtó. Ez nem mindig volt jellemző periféria, sőt, az olcsóbb gépekben még ma sem általános. A meghajtó hiánya telepítéskor igen kellemetlen, de nem áthidalhatatlan probléma. Ha legalább egy floppymeghajtóról el tudjuk indítani a telepítést, és van egy másik gép a közelben, melyhez ethernet-hálózaton csatlakozni tudunk, a telepítés megoldható. Ilyenkor egyetlen bootlemez helyett – disztribúciótól függően – 2–6 floppyra és esetleg sok kézimunkára lesz szükség. Ez utóbbi annak a következménye, hogy újabb és újabb PCMCIA eszközöket kapunk a géphez, és ezek sajnos nem kompatibilisek a régiekkel. A disztribúciókban megtalálható driverek csak ideigóráig számítanak a legújabbnak, hiszen a telepítőcsomagokat csak pár havonta adják ki, míg a PCMCIA driver jóval gyakrabban, néha hetek alatt frissül.

Hálózati telepítéshez szükség van egy másik gépre, amit kinevezünk szervernek. Ezen (akár CD-n, akár a merevelemezen) meg kell lennie a teljes telepítőnek, valamint egy NFS- vagy FTP-szervernek. Néhány disztribúció már SMB-ről (Microsoft Networking) is képes települni. A legjobb azonban az, ha a másik gép is linuxos, és NFS-ről telepítünk. Ehhez csak fel kell tenni az NFS-szervert (ez minden disztribúció része), ki kell tölteni a /etc/exports file-t (lásd 1. táblázat), melybe be kell írni a disztribúciót tartalmazó könyvtár nevét.

1. táblázat: minta /etc/exports file

```
/home 192.168.1.0/255.255.255.0
                    (rw,squash_uids=0-100,squash_gids=0-80)
/cdrom 192.168.1.0/255.255.255.0(ro)
```

A minta a /home és a /cdrom könyvtárakat kínálja fel a hálózatnak. Ha ez megvan, indítsuk el, vagy indítsuk újra az NFS-kiszolgáló programot. A telepítés közben be kell tölteni a PCMCIA vezérlő kezelőprogramját, mely szintén minden disztribúció része. Ez elindulása után magától megtalál minden eszközt. A hálózati csatoló beállítására később kerül sor, a telepítő szokásos hálózati telepítés beállításánál. Ezután már minden ugyanúgy megy, mintha egy asztali gépet telepítenénk a hálózatról. Ha NFS-ről telepítünk, az NFS-szervert szerver:/könyvtár formában kell megadnunk, kivéve, ha a telepítő külön-külön kéri ezeket az adatokat.

Advanced Power Management

Az APM a hordozható számítógépek legfőbb problémaforrása, de asztali gépeken is sokszor produkál spontán rendszerlefagyásokat. Ha ez előfordul a legtöbb amit tehetünk, hogy a BIOS-ban kikapcsoljuk –, bár néha ez sem elég. A telepítő által feltett Linux kerneleket szinte kivétel nélkül APM-támogatás nélkül fordították, ami jelen esetben nem felel meg nekünk. Még akkor is érdemes bekapcsolni az APM támogatást, ha a BIOS-ban kikapcsoltuk azt. Találkoztunk már olyan AST noteszgépekkel, melyek csak akkor nem fagytak le, ha azok BIOS-ában ki volt kapcsolva az APM, a Linux kernelben pedig be. Az AST válasza azon kérdésünkre, hogy hogyan lehetne életre kelteni az APM-et Linux alatt az volt, hogy használjunk Windowst.

Természetesen a fenti egy igen szélsőséges példa volt, de érzékeltetni szerettük volna a felmerülő problémákat és a nem ritka negatív hozzáállást a gyárak részéről. Hogy jó példát is említsünk, Portocom noteszgépen majdnem gond nélkül tudtuk használni az APM valamennyi funkcióját. Az

egyetlen amivel néha problémánk akadt, a Save to disk mód, azaz amikor a BIOS a teljes futó rendszert lemezre menti, majd onnan visszatöltve ott folytatja a futását minden program, ahol abbahagyta. Itt ritkán ugyan, de előfordult, hogy a visszatöltés után lefagyott a rendszer. Minden más altatásos leállítás hibátlanul működött, beleértve az elem állapotjelzőiét, és a gép kikapcsolását a leállításkor. A 2. táblázatban látható az általunk javasolt Linux kernel beállítása az APM használatához.

2. táblázat: javasolt Linux kernel beállítások

```
CONFIG_APM=y
# CONFIG_APM_IGNORE_USER_SUSPEND is not set
CONFIG_APM_DO_ENABLE=y
# CONFIG_APM_CPU_IDLE is not set
CONFIG_APM_DISPLAY_BLANK=y
CONFIG_APM_POWER_OFF=y
# CONFIG_APM_IGNORE_MULTIPLE_SUSPEND is not set
# CONFIG_APM_IGNORE_SUSPEND_BOUNCE is not set
CONFIG_APM_RTC_IS_GMT=y
# CONFIG_APM_ALLOW_INTS is not set
```

PCMCIA

A hordozható számítógépekhez általában minden lényeges külső eszközt PCMCIA csatolóval illesztünk. Mint a telepítésről szóló részben már utaltunk rá, ez nem mindig megy zökkenőmentesen. Vásárlás előtt érdemes tájékozódni, hogy mely noteszgépek PCMCIA csatolóját, és mely PCMCIA kártyákat támogatja a Linux. Ha feltettük a pcmcia-cs csomagot, a /usr/doc/pcmcia-cs/SUPPORTED.CARDS.gz file-ban megtaláljuk az összes támogatott kártya nevét. Jelen cikk írásakor a legújabb pcmcia-cs csomag a 3.0.9-es.

A PCMCIA eszközök használatához a fenti csomagon kívül a pcmcia-modules-<kernelverzió> csomagot is fel kell tenni, ahol a <kernelverzió> a futó kernel verziószáma, amellyel használni akarjuk a PCMCIA eszközt. A Debian előre lefordított kerneleihez számos ilyen csomag található a CD-n, de megtaláljuk a pcmcia-source csomagot is, melyből bármikor fordíthatunk magunknak új csomagot a Debian segédprogramjaival. Ha hálózatot is szeretnénk használni, ahhoz a Debian ad némi segítséget. Mivel a PCMCIA eszköz nem egy állandó csatoló, a hálózati beállítások nem kerülhetnek a /etc/init.d/network szkriptbe, helyette a /etc/pcmcia/network.opts file-t használja. Ezt a /usr/sbin/pcnetconfig paranccsal lehet beállítani, melynek indítását a PCMCIA-os csomag telepítésekor fel is ajánlja a rendszer. Ugyan a többi PCMCIA eszközhöz nincs ilyen beállító program, azok beállításai is a /etc/pcmcia könyvtárban találhatók, kivétel nélkül .opts végződéssel.

A grafikus felületű cardinfo program segítségével információkat kaphatunk egy-egy kártyáról, illetve kézzel is vezérelhetjük őket, ha felül akarjuk bírálni az automatikus kezelést.

Videokártya, képernyő

Az egyik legkényesebb rész, mely mindig egyedi beállítást igényel. Fontos odafigyelni vásárláskor, hogy legyen driver a kiválasztott gép videokártyájához az XFree86 csomagban. Ha nincs, még mindig választhatjuk az adott gépet, csak készüljünk fel arra, hogy az XFree86 helyett egy igen drága programot kell megvennünk a használatához. Az Accelerated-X-nek van külön noteszgép-változata, mely az ott használatos videokártyák nagy részét kezeli, de ez a program sokba kerül. Az általunk kipróbált Portocom noteszgépek szépen mentek az XFree86 megfelelő driverével, igaz ott is kézzel kellett beállítani pár dolgot, de ez már elsősorban az LCD sajátosságainak a következménye.

Mivel az LCD-képernyő kezelése teljesen más, mint a hagyományos monitoroké, azaz nem a frissítési frekvenciák határozzák meg a képet, hanem előre beállított fix felbontásban tud dolgozni, ennek megfelelően kell az X11-et beállítani.

Ha kisebb felbontást állítunk be mint az LCD legnagyobb felbontása, általában kisebb képet is kapunk. Itt is csak azt tudjuk tanácsolni, hogy az Interneten érdemes keresgélni a Linux laptop lapon. A fix felbontás miatt egyedi modeline kell a megfelelő felbontás eléréséhez, az XF86Setup és más beállító programok által készített XF86Config file modeline sorai ritkán jók. Ilyen modeline-okat az említett címen lehet találni, mi is ott találtuk meg a kipróbált noteszgépekhez megfelelőt.

További eszközök

Gyakran találkozunk IrDa-csatolóval és hangkártyával ezekben a gépekben. Az előbbi kezeléséhez a Linux IrDA-projekt által készített drivert érdemes használni, mely a 2.2-es Kernelben már megtalálható. Részletes információt, programokat a projekt honlapján lehet találni. A hangkártya megszólaltatásához pedig az ALSA (Advanced Linux Sound Architecture) meghajtóit javasoljuk. Ez is része a CD-mellékleten található Debian GNU/Linux disztribúciónak, és igen kellemes telepítővel van felszerelve. Természetesen sok esetben a kernelben található hangkártyadriverek használata is sikerrel járhat. ■

MUTATÓ

Linux laptop oldal:
www.cs.utexas.edu/users/kharker/linux-laptop/
Linux IrDa oldal: http://www.cs.uit.no/linux-irda/
ALSA oldal: http://alsa.jcu.cz/

Moldvai Dezső E.
mde@lme.linux.hu

Nyílt forrás – zárt hardver?

Új alkatrész vásárlásánál a műszaki jellemzőkön kívül fontos szempont az operációs rendszer általi támogatottság is, ami – különösen Linux esetén – problémát jelenthet. Ha megkérdezzük az eladótól, működni fog-e a kiválasztott részegység Linux alatt, sajnos ma még elég sok üzletben csak üveges tekintettel merednének ránk.

Valamivel jobb a helyzet, ha az eladó már hallott valamit a Linuxról, de ez még mindig messze van az ideális esettől, amikor biztosan meg tudja mondani, hogy az adott alkatrész működik, vagy – rosszabb esetben – sem, és nyugodtan fordulhatunk hozzá kérdéseinkkel az esetleg felmerülő gondokkal kapcsolatban.

Reméljük, hogy a nemrég megalakult Linux-felhasználók Magyarországi Egyesülete egyre több üzlettel tudja a Linuxot megismertetni, és támogatottságát növelni. Ez nem is olyan egyszerű feladat, ráadásul ez még csak a probléma kisebb része.

A hardvergyártók saját szempontjaik alapján választják ki az általuk támogatott operációs rendszereket, melyek közé eddig nem igazán került be a Linux.

Az elmaradt támogatásnak pedig a felhasználók látják kárát. A számítástechnika gyorsan változó piacán már nem egy gyártó kényszerült feladni a versenyt. Ekkor az általa nyújtott (biztosnak vélt) támogatás vagy teljesen megszűnt, vagy épphogy használható mértékűre csökkent. Szerencsére, néhányan segítőkészebben álltak a problémához, és (még idejében) elérhetővé tették a meghajtóprogramok írásához szükséges dokumentációkat.

Az annak idején remek választásnak tűnő MediaMagic hangkártyám egyértelműen illusztrálja a leírtakat: a gyártó megszűnésével az addig sem kitűnő driverek frissítése teljesen abbamaradt. A Windows 95 megjelenése után az egyik cég (mely nagy tételben építette be ezt a kártyát komplett gépeibe) kényszerült egy toldozott-foltozott, kizárólag midi-lejátszásra képes meghajtóprogram kiadására. Ma a hangkártya kitűnően, minden képességét kihasználva szól egy linuxos gépben.

Ez a példa azt is igazolja, hogy a szabad szoftverként terjesztett meghajtóprogramok sokszor jobb minőségűek a gyártóénál. Sajnos, a gyártók többségének csak az az érdeke, hogy a termékét megvegyék, míg a szabad szoftverfejlesztők annak minél jobb használatára törekednek.

Sokszor emlegetik még, hogy a „forrás a legjobb dokumentáció". Ez persze csak a megjegyzésekkel megfelelően ellátott kódra igaz, de mindenképpen segít bizonyos helyzeteket egyértelműsíteni.

A hangkártya esetének ellentéte az a Commodore MPEG Pro dekódoló kártyám, melynek támogatottsága egy Windows 3.1-es (hibákkal teli) driverrel ki is fújt, s a mai napig nem sikerült semmilyen információt szereznem róla, pedig nagyon szívesen használnám Linux alatt.

Ezek alapján felmerül a kérdés: a gyártók miért nem csak a fejlesztéssel foglalkoznak, miért nem teszik publikussá a programozási információkat, hogy esetleg jobb képességű programozók, jobb minőségű drivert készíthessenek?

Az információ eltitkolásának több lehetséges oka is van, ezért nem létezik néhány eszközhöz linuxos driver:

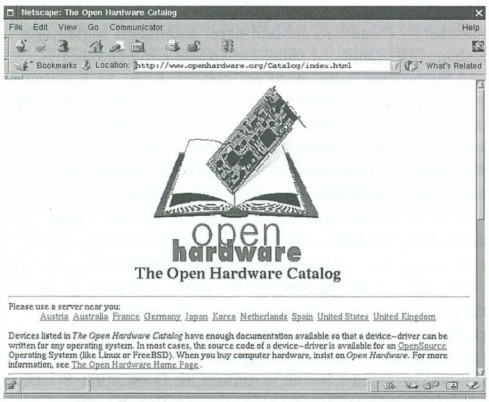
- Féltik azt a munkát és energiát (a pénzről nem is beszélve), melyet a fejlesztésbe fektettek, emiatt nem merik kiadni a leírásokat. A valóságban azonban egy termék lemásolásához (hamisításához) azok az információk nem ele-

gendőek, melyekkel már bőven lehet hozzá programot írni.

● A termékük valójában nem felel meg az általuk hirdetett követelményeknek (sem), a lespórolt alkatrészek feladatát a program a CPU-t használva valósítja meg, csökkentve a rendszer teljesítményét és a kompatibilitást (tipikus példája az úgynevezett winmodem).

● Kiadnák az információt, de csak a fejlesztőknek, akikkel NDA-t (titoktartási egyezmény) íratnak alá. Ezzel gyakorlatilag lehetetlenné válik a szabad forrású meghajtó készítése. A GPL miatt muszáj a módosított forrást elérhetővé tenni, míg az NDA pont ezt tiltja meg. Sokak szerint azonban jobb a semminél, ha csak bináris formában terjeszthető programokat készítenek (pl.: a Creative Labs fog ilyeneket kihozni).

● Ők maguk készítenek nyílt forrású drivert, azonban az úgy néz ki, hogy nem érvényes rá az Open Source meghatározás. A definíció szerint a kódnak olyan állapotúnak kell lennie, hogy azt a programozók módosítani tudják, hiszen nem mindegy, hogy egy konstans a szerepét meghatározó névvel, vagy csak az értékével szerepel. Ilyesmi történt ugyanis az nVidia által az XFree86 fejlesztőinek küldött forrással.



Az Open Hardware projekt címlapján a Linuxbarát gyártókról tájékozódhatunk

● Előfordul még egy különleges helyzet is, melyet a Geniusnál produkálnak: a nemrég vásárolt hálózati kártyám (GE2500 III) lemezén a Linux könyvtárban található file (hibásan, de lényegében érthetően) leírja, hogyan kell a kernelt és egyebeket beállítani a kártya működéséhez.

Ezek szerint tudnak a Linuxról és arról is, hogy a támogatásuktól mentesen fejlesztett driverrel működik a kártya, tehát „élvezik" annak előnyeit.

Ugyanakkor a Genius EasyScroll egeremmel kapcsolatos kérésre azt a választ kaptam, hogy örülnének a linuxos driver megírásának, és amint aláírok egy NDA-t, küldik is a dokumentációt.

Ezzel egyértelműen ki is zárták, hogy a gpm-be bekerüljön a támogatás. Végül a soros port figyelésével sikerült rájönni az egér által használt nem szabványos protokollra, így a következő gpm valószínűleg már támogatni fogja az eddigi három gombos módon kívül a görgőt és az oldalsó gombot is, mint ahogyan más görgős egerekkel (pl.: LogiTech Cordless Wheel Mouse) teszi.

Szerencsére léteznek olyan gyártók is, melyeket a fentiek nem riasztanak el a szükséges információk nyilvánossá tételétől.

Ezek összefogására jött létre az Open Hardware Certification Program, mely ígéretet kér a gyártótól, hogy – meghatározott feltételek mellett – hozzáférhetővé teszi a szükséges dokumentációkat.

Érdekes módon megdöbbentően kevés termék kapta meg az elismerést és a vele járó logót (azért ennél jóval többen teszik közzé a leírásokat, csak éppen nem ígérnek semmit a jövővel kapcsolatban).

Az alacsony szám feltehetően az alul-reklámozottságnak köszönhető. Annak ellenére, hogy az egyik fő támogatón, a Linux Internationalen keresztül olyan cégek állnak mögötte, mint például a Compaq, Sun, SGI és az IBM, senki sem ír a programról.

Egyre többen ismerik fel a Linuxban rejlő lehetőségeket, és ezzel a nyílt rendszerek filozófiája is beszivárog a köztudatba. Remélhetőleg az új hozzáállás nemcsak a dokumentációk nyitottabb kezelésében nyilvánul meg, hanem a kereskedők és a vásárlók viszonyát is új alapokra fogja helyezni. ■

MUTATÓ

http://www.openhardware.org
http://www.linux.org

Pásztor György
pasztor@linux.gyakg.u-szeged.hu

Terminálépítés

Sokakban felmerül az igény, hogy elavult gépeiket felhasználják valamilyen célra. A Unixok világában az X Windows vagy X11 nemcsak erre a problémára ad választ, hanem azt is lehetővé teszi, hogy korszerű PC-inket grafikus munkaállomásként állítsuk hadrendbe.

Mindehhez csak egy erős gép kell, amelyen az alkalmazásokat futtatjuk. Ehhez kapcsolódik a hálózaton keresztül a többi grafikus munkaállomás, melyek gyakorlatilag féligmeddig lebutított gépek. Egy ilyen konzolhoz billentyűzet, egér, monitor és jobb esetben valamilyen hangkeltő eszköz kapcsolódik.

Megfelelő összegekért kész terminál-megoldásokat is kaphatunk, melyeknek a hardvere eleve csak erre a célra készült. Azonban sokkal jobban járunk, ha már meglévő, szinte tetszőleges típusú gépeinket alakítjuk terminállá a Linux segítségével. E megoldás pénzügyi szempontból igen kedvező – ingyenes.

Ma a boltban kapható, vagy az Internetről ingyen letölthető Linux-disztribúciók mindegyikében megtalálható az X Windows felület. Az alapoknál kezdve, itt kicsit mást jelent az X-kiszolgáló (X-Server) és az X-ügyfél (X-kliens), mint ahogyan azt gondolnánk.

● Az X-kiszolgáló az a program, amely a felhasználó gépén (nevezzük X-terminálnak) fut. Ez a program jeleníti meg az ablakokat a képernyőn, és közvetlen kapcsolatban áll az egérrel, a billentyűzettel, ez küldi tovább a konzolról vett bemenetet az X-ügyfélprogramnak.

Ezen programok ingyenes és legújabb változatát az XFree86 szervezet weblapjáról töltheti le: http://www.xfree86.org/.

● Az X-ügyfél az a program, amivel gyakorlatilag dolgozunk, és a szervergépen fut. Ez lehet például egy szövegszerkesztő, táblázatkezelő, irodai programcsomag vagy akár egy hálózati böngésző is. (Megjegyzés: A Netscape-et például többfajta Unix operációs rendszerre is elkészítik–elkészítették, többfajta processzorra, és természetesen X11 támogatással.) Ez az a program, amely az X-kiszolgálónak küld, és attól fogad üzeneteket. Az üzenet jelezheti, hogy lenyomtak egy billentyűt az X-terminál konzolján, vagy az alkalmazás egyik ablakának mire változott meg a tartalma. Ez eddig szép, de ez még csak a kép volt. A hang továbbítására is ugyanilyen lehetőség nyílik. Ennek neve: NAS (Network Audio System). Ez gyakorlatilag az X Windows párja–kiegészítője. Természetesen itt is az a program a kiszolgáló (Audio Server), amelyik azon a gépen fut, ahol a hangokat halljuk, és az a program az ügyfél, amelyik azon a gépen fut, amelyen a hanglejátszó program tevékenykedik. (Természetesen mikrofon is kapcsolódhat az X-terminálunkhoz. Ilyenkor az AU-Server továbbítja a digitalizált hangot, és az AU-ügyfél pedig rögzíti egy file-ba.)

A CHIP Magazinban már több Linux-disztribúciót is közreadtunk. Ezen túlmenően a legújabb fajta monitorvezérlő kártyákhoz is folyamatosan megtalálhatók a legfrissebb X-kiszolgáló programok. A terminálgépekre nem nehéz telepíteni a Linuxot, ennek részleteire nem térek ki, hiszen a CHIP Magazinban több írás is megjelent az installálásról és a grafikus felület beállításáról. (Kiadványunkban ezzel a témával foglalkozik a Debian GNU/Linux 2.1 (Slink) telepítése című cikk is.)

A fentiekben sehol nem említettem, hogy az X-terminálokban lenne bármilyen háttértár. Ez nem véletlen, tudniillik általában nem szokott lenni. Ha mégis van, akkor azt leginkább a fizikai memória kibővítésére (háttér cserefile/swapfile) érdemes használni. Az X-terminálok készítésének leírásánál nem is feltételezem a háttértár meglétét.

A Unixok hálózati állománymegosztását NFS-nek (Network File System) hívják. A Linux-kernelnek van egy root-on-nfs nevű lehetősége, ami azt jelenti, hogy a gyökér (root) könyvtárat tartalmazó file-rendszer fizikailag nem a mi gépünkben található, hanem a hálózatról érhető el, mégpedig NFS segítségével. Az X-terminálnak viszont valahonnan meg kell tudakolnia, hogy hol van ez a hálózaton. E cél megvalósításához két fajta protokollt is találunk a Linux-kernel opciói között: a BOOTP-t és RARP-t. Ez utóbbit csak kompatibilitási okokból hagyták benne a kernelben, és a későbbi verziókban már nem biztos, hogy szerepel.

A bootp beállítása

Egy bootp szerver konfigurációja nem ördöngösség, mint az 1. listán látható példa is mutatja.

```
# Példa bootptab fájl (domain=valahol.hu)
            .default:\
                  :hd=/pub:bf=null:\
                  :ds=10.0.0.2:\
                  :ns=10.0.0.2:\
                  :sm=255.0.0.0:\
                  :ts=10.0.0.2:\
```

```
                :ht=ethernet:\
                :gw=10.0.0.2:\
                :hn:to=3600:
gep1:ha=00A024C7B2F7:ip=10.0.1.1:rp=/X/10.0.1.1/:tc=.default:
gep2:ha=00A024CA8621:ip=10.0.1.2:rp=/X/10.0.1.2/:tc=.default:
```

A konfigurációs állományt /etc/bootptab néven kell tárolni.

Némi magyarázat a táblázat kitöltéséhez: az egyes sorok elemei kettősponttal vannak elválasztva, és a sort is kettősponttal zárjuk le. Ha valami nem fér ki egy sorba, akkor használhatjuk a \ (balra dőlő vonal) jelet, hogy a következő sorban ugyanazt a bejegyzést tudjuk folytatni. Talán említenem sem kell, hogy minden sor egy bejegyzés, és ha megjegyzéssort akarunk betenni, akkor azt egy # (kettős kereszt) jellel kezdődő sorban tehetjük meg.

- A file elején .defaulttal bevezettem egy olyan részt, amely közös elemeket tartalmaz minden egyes gépre nézve. Természetesen nem kötelező ezt .defaultnak hívni, és nem csak egy ilyen lehet. Ha több különböző beállítással rendelkező gépnek kell bootp-választ adnunk, akkor a gépek minden csoportjának felvehetünk egy újabb csoportot, újabb néven. Ezt a részt teljesen a bootptab man oldalának megfelelően töltöttem ki.
- hd: Ezzel annak a könyvtárnak a nevét adhatjuk meg, amelyben az úgynevezett bootfile található. Ez azon kliensek számára fontos, amelyekben nincs merevlemez, vagy más olyan eszköz, amiről bootolni tudna, viszont a hálózati kártyában olyan boot-EPROM található, amely képes a bootp protokoll segítségével a rendszert indítani.
- bf: Ezzel az elemmel azt jelölhetjük ki, hogy melyik az a file, amit boot-image-nek használunk az adott klienshez. (Ebben az esetben nem bootp-vel bootoltak a kliensek, csak a példa kedvéért vettem bele a leírásba.)
- ds: Ide a Domain Name Serverek listáját írhatjuk be, vesszővel elválasztva.
- ns: Itt az IEN-116 name-szerverek címeinek listája adható meg. (Nem valószínű, hogy szükség lenne erre az adatra, de ha mégis, akkor tudjuk, hogy hol keressük.)
- sm: A kliens gép (az az X-terminál, aki a választ kapja) subnetjének a netmaskja. Ez szintén hálózati adat, amit a helyi hálózat rendszer-adminisztrátorának feladata ismerni.
- ts: A használni kívánt time-szerverek címeinek listája (szintén veszővel elválasztva).
- ht: A kliensgép hardverének típusa: ehhez megfelelő számokat a megfelelő RFC-ben találunk, de nem szükséges feltétlenül számokat használnunk, használhatjuk helyette, az ethernet vagy ether, ethernet3 vagy ether3, iee802, tr vagy tokenring, arcnet stb. rövidítések valamelyikét is. (Részletek: man bootptab.)
- gw: Ide a gateway-ek címeinek listáját írhatjuk be. (Szintén hálózati adat.)
- hn: Ezzel jelezhetjük, hogy szeretnénk, ha a bootp-server a kliensnek a címe mellett a nevét is megmondaná. (Azt, ami a bootptab-ban szerepel az egyes sorok elején.)
- to: A helyi időeltolódás az UTC-hez képest másodpercekben.

Ezek olyan beállítások voltak, amelyek általában a gépek nagyobb csoportjára jellemzők. Azonban ezek közül sem kötelező az összes adatot megadni, csak azokat, amelyek a kliens elindításához feltétlenül szükségesek Most következzenek azok, amelyek általában egyéniek.

- név: Minden egyes kliensről szóló bejegyzést a nevével kezdek – hasonlóan a csoportok leírásához. Ott a pont után a csoport neve állt; itt nem szerepel pont, és csak egyetlen gépről van szó.
- ha: A kliens hálózati kártyájának a címe. Ilyen címe minden egyes hálózati kártyának van, és a világ minden hálózati kártyájának különböző a beégetett címe. Ezt ne tévesszük össze az IP-címmel, amit mi tudunk kiosztani.
- ip: A kliensnek szánt IP-cím.
- rp: Ha a kliensünk root-on-nfs-t használ, akkor a bootp és az NFS-szerver általában egy gép. Ilyenkor ezzel a bejegyzéssel tudjuk megadni, hogy a kernel melyik könyvtárat mountolja fel az NFS-szerverről gyökérkönyvtárnak. Azt, hogy hogyan kell az NFS-szervert beállítani, az alábbiakban írom le.
- tc: Az itt kijelölt csoport beállításaival egészülnek ki a gép adatai. A példában a .default csoportból vettem a közös elemeket, például a netmaskot, a name-szervert, a gateway-t, a hardvertípust stb.

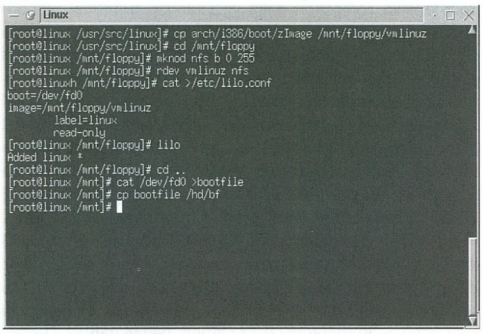
Ezek azok az elemek, amelyek talán a legsűrűbben szerepelnek (de a bootptab man-ja által írt példában mindenképpen). Talán még a td-ről tennék említést, amely a tftp-boot-EPROM-os gépeknél használatos. A tftp-szerverekhez ezzel lehet megadni a gyökérkönyvtárat. Ilyenkor alapesetben a bootfile, és egyéb dolgok kiválasztása az IP-cím alapján történik, de minden részletet megtalálunk a manban. Ha ingyenes boot-EPROM-kódra van szükségünk a sunsite.unc.edu-ról lehet letölteni a Netboot-nfs és az etherboot nevű bootprogramokat (a netboot benne van a CD-mellékleten megtalálható Debian Linuxban – a szerk.), azonban ezek használata előtt mindenképpen kérdezzünk meg tapasztaltabbakat, nehogy félreégessük a boot-EPROM-ot.

Most jutottunk el odáig, hogy a klienseink be tudnának bootolni, ha valamilyen módon elindítanánk rajtuk egy megfelelően konfigurált Linux-kernelt. Ahhoz, hogy a klienseken használható kernelt kapjunk, a következő dolgokat mindenképp bele kell fordítani. Nem modulba, hanem bele a kernelbe, mert már a bootfolyamat során szükség van rájuk.

- Az általános hálózati támogatást.
- A TCP/IP-hálózat támogatását.
- A kliens gépben lévő hálózati kártya vezérlőprogramját.
- Az NFS-file-rendszert, a root-on-nfs opcióval és a bootp-protokollal együtt.

Fogjunk egy lemezt, formázzuk meg (lehetőleg ext2fssel). Másoljuk rá a kernelt. Hozzunk létre egy nfs nevű eszközt a

példa szerint (1. kép), majd tegyük ezt az eszközt a kernelünk rooteszközévé az rdev paranccsal. Futtassuk le a LILO-t az alább írt konfigfile-lal. Ha mindezzel megvagyunk, készítsük el a lemezről az image-et, és ennek kell működnie bootfile-ként. (Ennél egyszerűbb, ha a már említett netboot csomagból használjuk az mknbi-linux programot, mely egy Linux kernelfile-ból tftp-vel vagy floppyról indítható formátumot csinál – a szerk.) Azért nem árt előbb egy kliensen kipróbálni a bootlemezt, hogy elindul-e vele a gép, és megpróbálja-e a bootp-szervert keresni. Továbbá felismeri-e a hálózati kártyát. De természetesen nem ez az egy út van egy X-terminál elindítására. Ha például DOS bootol automatikusan (egy NetWare-szerverről), és emiatt nem akarunk boot-EPROM-ot cserélni, használhatjuk a loadlint, de ebben az esetben is futtassuk le az mknodot és az rdevet, mert ezzel állítjuk be úgy a kész kernel image-t, hogy a bootp-t és a root-on-nfs-t használja rooteszköznek. Ezután csak át kell másolni a file-t olyan helyre, hogy a DOS-os kliensről is elérhető legyen, és ki kell adni a loadlin vmlinuz parancsot.



Bootolható image készítése

Megjegyzés: a hd, és bf helyére helyettesítsük a bootptabban beírt adatokat.

NFS, de nem Need for Speed

Most már, ha az NFS-szerver is működne, és be lenne állítva, akkor megpróbálna betöltődni a rendszerNFS-ről. Azért a biztonság kedvéért – még, ha nem is állítottuk még be az NFS-szervert –, próbáljuk ki, hogy eddig minden működik-e! Ne a végén kezdjük a hibát keresni, mivel így sok bosszúságtól óvhatjuk meg magunkat. Ne az utolsó pillanatban vegyük észre például az olyan bakit, hogy az ipfwadm-mal letiltottuk azt a lehetőséget, amely szükséges lenne ahhoz, hogy a szerver bootp válaszolni tudjon a klienseknek. Következő lépésként jöjjön az NFS-szerver konfigurálása. Ez sem nagy ördöngösség: a /etc/exports file-ba kell beírni azokat a könyvtárakat, amelyeket közzé akarunk tenni, és írjuk be ide azt is, hogy mely gép számára tesszük elérhetővé őket.

Erre mintát a 2. listán láthatunk.

```
/usr 10.0.0.0/255.0.0.0 (ro,squash_uids=0-100,squash_gids=0-
80)
#1es gép
/X/10.0.1.1 10.0.1.1(rw,no_root_squash,link_absolute)
#2es gép
/X/10.0.1.2 10.0.1.2(rw,no_root_squash,link_absolute)
Egy példa arra, hogy hogyan tegyük elérhetővé mások számára a
szerverünket nfs-en.
```

A sorok formátuma: egy exportálandó könyvtár neve, majd a gépek amelyek elérhetik, és zárójelben az esetleges opciók. A fenti példából látható, hogy egész alhálózatokat is megadhatunk a gépek címzésénél, de az nem derül ki, hogy nemcsak IP-szám, hanem gépnév is megadható. A fontosabb opciók, amelyekre szükségünk lehet:

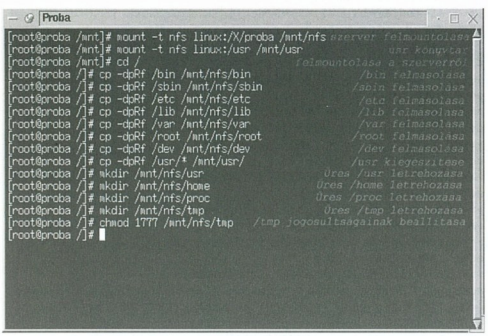
- ro: Read-Only: Csak olvashatja az adott területet a kliens.
- rw: Read-Write: Írhatja és olvashatja is az adott területet a kliens.
- link_absolute: nem túl fontos opció, de nekem – ezt használva – sohasem volt problémám. A szimbolikus linkeken nem változtat az NFS-szerver. Egyébként ebben az esetben ez az alapbeállítás is. Viszont néha jó trükk kiírni az alapbeállításokat, ha valamivel kapcsolatban felmerült valamilyen problémánk. Később, ha újra elölről kell kezdeni, akkor így eszünkbe jut róla minden, és nem kell az elejéről olvasni a man és HOWTO oldalait.
- no_root_sqash: Szándékosan hagytam a legvégére a legfontosabb opciót, amely az úgynevezett diskless kliensek esetén kritikus. Az opció azt befolyásolja, hogy a szerver adjon-e root jogot, ha a kliens azt kér. Normális esetben az NFS-szerver a 0-ás, azaz root user jogot kérő kliensnek „nobody" jogot ad, azaz ha a kliensen root-ként dolgozva a szerverről származó file-rendszeren létre akarunk hozni valamit, akkor a szerver szemszögéből a nobody userrel tesszük ezt. Ez néha nem jó, és ezt hivatott megváltoztatni ez az opció. A szerver ezen területét tartalmazó file-rendszert érdemes „nosuid,nodev" opciókkal mountolni, hogy a kliensről ne lehessen a szerveren is érvényesülő setuid és device file-okat létrehozni.

Jelenleg ott tartunk, hogy van egy bootp-szerverünk, és ugyanezen a gépen NFS-szolgáltatás is fut, csak éppen még nem tettünk rá semmit, amit érdemes lenne kiszolgálni. Ezért most megtöltjük tartalommal az NFS-szerverünket. Itt most feltételezem, hogy be tudunk szerezni egy ideiglenes háttértárat valamelyik X-terminálunkba. Ha beszereltük, akkor telepítsünk rá valamilyen tetszőleges Linux-disztribúciót. Természetesen hálózat, X11, hangkártya, NAS-támogatással. Az utóbbi kettőre abban az esetben van szükség, ha

zenét is szeretnénk hallgatni X-terminálja inkon. Még abban az esetben is érdemes felrakni, ha tudjuk, hogy az adott X-terminálon éppen nincs hangkártya, de várható, hogy később gépeink egyikében lesz, és szeretnénk majd szóra bírni. Készült már olyan kernelpatch is, amely hangkártya helyett a PC speakerét használja hangkeltő eszközként. Ha a csipogóval szeretnénk hangoskodni, akkor is érdemes feltelepíteni a NAS-t.

Az NFS-szerverről mountoljuk fel azt a könyvtárat, amit ideiglenesen ennek a barkácsolt gépnek szánunk. Majd másoljuk fel a /bin, /sbin, /etc, /lib, /var, /root, /dev könyvtárakat az NFS-szerverre, valamint hozzunk ott létre egy /tmp könyvtárat, ugyanazokkal a jogosultságokkal, mint az eredetié, ezek mellett egy-egy üres /usr, /home és /proc könyvtárra is szükségünk lesz. Most feltételeztem, hogy az NFS-szerverünkön több minden fel van installálva, mint ami egy X-terminál klienséhez szükséges. Ha nem így lenne, akkor az NFS-szerver /usr könyvtárát bővítsük ki a hiányzó részekkel. Ebben az esetben ne felejtsük el ideiglenesen beállítani a /etc/exportsban, hogy az adott gép írhassa is a /usr könyvtárat, és azt se felejtsük el valamilyen más nevű könyvtár alá ideiglenesen felmountolni a kliensről.

```
Kliens (X-trminál) Szerver (NFS,BOOTP) 10.0.1.1 (proba)
10.0.0.2 (linux)
/mnt/nfs ----> /X/proba /mnt/usr ----> /usr
```



A próbagép elkészítésének lépései

Ha ezzel készen vagyunk, akkor keressük meg a /etc/inittab-ban, hogy milyen szkripteket indít el a automatikusan a rendszer a bootolás után. (Keressük meg a sysinit bejegyzést tartalmazó sort!) Ezt a részt azért nem írom le konkrétan, mert attól a disztribúciótól függ, amelyet telepítettünk. Ha megvannak az initszkriptek, akkor bogarásszuk ki azt a részt, amely a gyökérfile-rendszert ellenőrizné, és kommentezzük ki, mert csak bajt okoz, ha az fsck megpróbál NFS-es file-rendszert ellenőrizni. Ha nincs az X-terminálok háttértáráról (már ha van nekik!) felmountolandó dolog, akkor nyugodtan megjegyzésbe tehetjük ezt a részt is, de inkább a /etc/fstabot állítsuk be igényeink szerint. Az X-terminál grafikus felületét ne felejtsük el konfigurálni, még mielőtt a /etc könyvtárat felmásolnánk! Nem kell mindent konfigurálni benne. Egy X-terminál grafikus felületéből csak annyi kell, hogy rendesen elinduljon a grafikus üzemmód, és lássa az egeret. Ez gyakorlatilag annyit jelent, hogy helyesen be legyen állítva a /etc/X11/XF86Config nevű file a kliensen.

A grafikus bejelentkeztető programot, az ablakkezelőt és a grafikus felhasználói programokat majd egy alkalmazásszerverről fogjuk futtatni. Nem kötelező, de nem is tilos, hogy az alkalmazásszervernek kinevezett, tehát a grafikus felhasználói programokat futtató gép nyújtsa az nfs+bootp szolgáltatást. A választást általában anyagi lehetőségeink szabják meg. A különválasztásnak inkább biztonsági okai vannak, a hálózati betörőkkel szemben bizonyos fokú védelmet jelent, ha két külön szerver van. Ugyanis, ha bootp+nfs-szervert feltörik még nem vesznek oda a felhasználók adatai. Ráadásul az X-terminálok kiszolgálásához az NFS-szerverről elég egyetlen backup, míg a felhasználók adatait nem lehet elég sűrűn menteni, hogy mindig a legfrissebb adatokat tudjuk visszaállítani egy esetleges katasztrófa esetén.

Újgyakorlatok egy kliensadminisztrációra

Végre megtöltöttük tartalommal az NFS-szervert, de ezek még csak egyetlen kliens file-jai, és fárasztó lenne minden gép miatt külön-külön felmásolni őket. Ráadásul ez iszonyú nagy pazarlás. Ehelyett alkalmazhatjuk a hard-linkelést, és így csak egyszer foglalják a helyet a különböző kliensek egyforma file-jai. Ez főleg a futtatható file-oknál térül meg, mert az szinte minden kliens esetében egyforma, és ha van is bizonyos eltérés a kliensek közt, akkor az általában minimális. Erre az esetre javaslom a következő megoldást. Amennyiben az IP-számokat szabadon oszthatjuk ki, akkor a gépek hardvere szerint egy- vagy kétszintű hierarchiát állítsunk elő. Az alábbi IP-címtartományokkal mindenki szabadon rendelkezhet (IP-Subnetworking-HOWTO alapján):

- A-osztályú tartomány: 10.0.0.0 (netmask: 255.0.0.0)
- B-osztályú tartományok: 172.16.0.0-172.31.0.0 (netmask: 255.255.0.0)
- C-osztályú tartományok: 192.168.0.0-192.168.255.0 (netmask: 255.255.255.0)

Ezeket elméletileg semmilyen, az RFC-ket betartó router nem továbbítja sehová, így kiválóan megfelelnek a belső

hálózat X-termináljai számára. Ezekkel egy A-osztályú tartománynál kétszintű hierarchiát biztosíthatunk a gépeinknek, egy B-osztályú tartománnyal pedig egyszintűt. Egy C-osztályú tartománnyal azonban nem tudunk hierarchiát kialakítani az IP-számokkal. A B-osztályú tartományból nem egy, hanem 16; C-osztályúból pedig 256 áll a rendelkezésünkre. Ezek abban az esetben hasznosak, ha a klienseink számára nincs elég IP-címünk, vagy egyszerűbben szeretnénk karbantartani őket. Így ugyanis, ha jól szervezzük a hálózatot, akkor ki tudjuk találni az IP-számból, hogy hol van a gép, és milyen hardverelemekből áll. Ha a 10.0.0.0-ás tartományt használjuk, akkor megtehetjük, hogy a 10.x.y.z IP-című gépnél az x-szel utalunk arra, hogy a gép melyik épületben/teremben van, az y-nal pedig arra, hogy milyen hardvert tettünk bele. Ráadásul egy terem egyforma gépeit is el tudjuk 255 sorszámmal látni, ugyanis nem valószínű, hogy egy teremben lenne 200 egyforma gép. Nem kötelező az általam javasolt IP-címkiosztást követni, ízlés dolga, hogy ki mi szerint sorszámozza a gépeit. Csak akkor érdemes energiát fektetni a hierarchiába, ha nincs elég IP-címünk, vagy ha sok gépünk van.

Ott tartottunk az NFS-szerverrel, hogy már volt egy gép, aminek az állományai készen fel voltak másolva a kiszolgálóra. A klienseket nem kell külön-külön létrehozni. Elég egy kis leleményes shellszkriptet írni, ami létrehozza a kliensek gyökérkönyvtárait az NFS-szerveren. Egy ilyen programot láthatunk a 4. listán.

Egy rövid kis script annak leellenőrzésére, hogy létezik-e egy fájl. Akkor hasznos, amikor meg akarjuk vizsgálni, hogy egy könyvtár üres-e.

Egy kisebb script arra, hogy hatékonyan egy paranccsal létrehozzunk újabb gépek root könyvtárait az NFS-szerveren.

```
#!/bin/bash
[ -e <1 ]||exit 1
#!/bin/bash
#az összes kliens alapkönyvtara
cdir="/X"
#az ip-k elso jegye
bip="10"
#Az nfs server neve
server="10.0.0.2"
#domainnév
dname="gyakg.u-szeged.hu"
#kliensek gatewaye,netmaskja,networkje,broadcastje
clgw="<server"
clnetmask="255.0.0.0"
clnetwork="10.0.0.0"
clbcast="10.255.255.255"
echo „Megkezdem <bip.<1.<2.<3 letrehozasat"
checkexisting <cdir/<bip.<1.<2.<3 || mkdir
<cdir/<bip.<1.<2.<3
echo „Alapallomanyok bemasolasa"
checkexisting <cdir/basedirc/* && cp -d -p -R -f
<cdir/basedirc/* <cdir/<bip.<1.<2.<3/
echo „Specallomanyok bemasolasa"
checkexisting <cdir/basedirc.<1.<2/* && cp -d -p -R -f
<cdir/basedirc.<1.<2/* <cdir/<bip.<1.<2.<3/
echo „Alapallomanyok belinkelese"
checkexisting <cdir/basedir/* && cp -d -l -p -R -f
<cdir/basedir/* <cdir/<bip.<1.<2.<3/
echo „Specallomanyok belinkelese"
checkexisting <cdir/basedir.<1.<2/* && cp -d -l -p -R -f
<cdir/basedir.<1.<2/* <cdir/<bip.<1.<2.<3/
echo „Create fstab"
if [ -f <cdir/<bip.<1.<2.<3/etc/fstab ]
then
 rm -f <cdir/<bip.<1.<2.<3/etc/fstab
fi
echo „#" >><cdir/<bip.<1.<2.<3/etc/fstab
echo „# /etc/fstab" >><cdir/<bip.<1.<2.<3/etc/fstab
echo „#" >><cdir/<bip.<1.<2.<3/etc/fstab
echo „#" >><cdir/<bip.<1.<2.<3/etc/fstab
echo „# <device> <mountpoint> <filesystemtype> <options>
<dump> <fsckorder>" >><cdir/<bip.<1.<2.<3/etc/fstab
echo „" >><cdir/<bip.<1.<2.<3/etc/fstab
echo „none /proc proc
defaults">><cdir/<bip.<1.<2.<3/etc/fstab
#egyéb nfs-szervereket is felmountolhatunk,
#mint például a szegedi linuxos anyagokat
#echo „sol.cc.u-szeged.hu:/pub/ftp-data/pub/linux /mnt/sol
nfs ro,auto">><cdir/<bip.<1.<2.<3/etc/fstab
echo „<server:/usr /usr nfs
ro,auto,rsize=16384">><cdir/<bip.<1.<2.<3/etc/fstab
echo „<server:/home /home nfs
rw,auto,rsize=4096,wsize=4096">><cdir/<bip.<1.<2.<3/etc/fstab
echo „<server:<cdir/<bip.<1.<2.<3/home/kliens/ /home/kliens
nfs
rw,auto,rsize=16384,wsize=16384">><cdir/<bip.<1.<2.<3/etc/fst
ab
if [ -f <cdir/basedir/etc/debian_version ]
then
echo „Create hostname"
# uj hostnamebeallitas: ha van akkor /etc/hosts alapjan
x=`cat /etc/hosts|grep -w <bip.<1.<2.<3|awk '{ print <2 }'`
x=<{x:-"<bip.<1.<2.<3"}
echo „<x" ><cdir/<bip.<1.<2.<3/etc/hostname
echo „Create init.d/network"
if [ -f <cdir/<bip.<1.<2.<3/etc/init.d/network ]
then
  rm -f <cdir/<bip.<1.<2.<3/etc/init.d/network
fi
echo „#! /bin/sh" >><cdir/<bip.<1.<2.<3/etc/init.d/network
echo „ifconfig lo 127.0.0.1"
>><cdir/<bip.<1.<2.<3/etc/init.d/network
echo „route add -net 127.0.0.0"
>><cdir/<bip.<1.<2.<3/etc/init.d/network
echo „IPADDR=<bip.<1.<2.<3"
>><cdir/<bip.<1.<2.<3/etc/init.d/network
echo „NETMASK=<clnetmask"
>><cdir/<bip.<1.<2.<3/etc/init.d/network
echo „NETWORK=<clnetwork"
>><cdir/<bip.<1.<2.<3/etc/init.d/network
echo „BROADCAST=<clbcast"
>><cdir/<bip.<1.<2.<3/etc/init.d/network
echo „GATEWAY=<clgw" >><cdir/<bip.<1.<2.<3/etc/init.d/network
echo 'ifconfig eth0 <{IPADDR} netmask <{NETMASK} broadcast
<{BROADCAST}' >><cdir/<bip.<1.<2.<3/etc/init.d/network
echo 'route add -net <{NETWORK}'
>><cdir/<bip.<1.<2.<3/etc/init.d/network
echo '[ „<{GATEWAY}" ] && route add default gw <{GATEWAY}
metric 1' >><cdir/<bip.<1.<2.<3/etc/init.d/network
chmod +x <cdir/<bip.<1.<2.<3/etc/init.d/network
elif [ -f <cdir/basedir/etc/redhat-release]
echo „Create sysconfig/network"
if [ -f <cdir/<bip.<1.<2.<3/etc/sysconfig/network ]
then
 rm -f <cdir/<bip.<1.<2.<3/etc/sysconfig/network
fi
# uj hostnamebeallitas: ha van akkor /etc/hosts alapjan
x=`cat /etc/hosts|grep -w <bip.<1.<2.<3|awk '{ print <2 }'`
x=<{x:-"<bip.<1.<2.<3"}
echo „NETWORKING=yes"
>><cdir/<bip.<1.<2.<3/etc/sysconfig/network
```

```
echo „HOSTNAME=<x" >><cdir/<bip.<1.<2.<3/etc/sysconfig/net-
work
echo „DOMAINNAME=<dname" >><cdir/<bip.<1.<2.<3/etc/syscon-
fig/network
echo „GATEWAY=<clgw" >><cdir/<bip.<1.<2.<3/etc/sysconfig/net-
work
echo „GATEWAYDEV=eth0"
>><cdir/<bip.<1.<2.<3/etc/sysconfig/network
echo „Create sysconfig/network-scripts/ifcfg-eth0"
if [ -f <cdir/<bip.<1.<2.<3/etc/sysconfig/network-
scripts/ifcfg-eth0 ]
then
rm -f <cdir/<bip.<1.<2.<3/etc/sysconfig/network-
scripts/ifcfg-eth0
fi
echo „DEVICE=eth0">><cdir/<bip.<1.<2.<3/etc/sysconfig/net-
work-scripts/ifcfg-eth0
echo „IPADDR=<bip.<1.<2.<3">><cdir/<bip.<1.<2.<3/etc/syscon-
fig/network-scripts/ifcfg-eth0
echo „NETMASK=<clnetmask">><cdir/<bip.<1.<2.<3/etc/syscon-
fig/network-scripts/ifcfg-eth0
echo „NETWORK=<clnetwork">><cdir/<bip.<1.<2.<3/etc/syscon-
fig/network-scripts/ifcfg-eth0
echo „BROADCAST=<clbcast">><cdir/<bip.<1.<2.<3/etc/syscon-
fig/network-scripts/ifcfg-eth0
echo „ONBOOT=yes">><cdir/<bip.<1.<2.<3/etc/sysconfig/network-
scripts/ifcfg-eth0
echo „BOOTPROTO=none">><cdir/<bip.<1.<2.<3/etc/sysconfig/net-
work-scripts/ifcfg-eth0
then
else
echo „Nem ismerem a kliensed disztribucióját"
fi
```

Ez a kis szkript nagyon hasznos, és könnyen módosítható, ha a klienseket hierarchiába szerveztük. Használata egyszerű: mkclient x y z, ahol az x y és z a kliens IP-címének utolsó három számjegye. Egy basedir nevű könyvtárba kell tenni azokat az állományokat, amelyek közösek lehetnek minden kliensen, ilyenek például a /bin, /sbin, és a /lib. Azokat a file-okat, amik a kliensek különböző csoportjainál különbözők lenhetnek, de nem kell, hogy egyéniek legyenek minden kliensen (tehát linkelhetők), a basedir.x.y nevű könyvtárakba kell tenni, és egy 10.x.y.z című kliens esetén ezekből veszi a szkript a szükséges különböző beállításokat. Erre jó példa, hogy ha valaki az X-szervert a /etc/X11/X paranccsal indítja. Ilyenkor csak a /etc/X11/X file-t kell megváltoztatni, ami általában egy szimbolikus link a /usr/X11R6/bin/XF86_<server>-re, és a különböző grafikus kártyákkal rendelkező gépeken a sysmlinkek csak egyszer foglalják a helyet. Igaz, ez nem nagy megtakarítás, de ha az egyik helyen megváltoztatjuk, akkor legalább a többi követi. Továbbá, ha ugyanezt az /etc/X11/XF86Config-filelal tesszük meg, akkor már nyerünk is néhány byte-ot. Ugyanez a séma működik a basedirc nevű könyvtárakkal is, a különbség mindössze annyi, hogy itt a file-okat nem linkeli, hanem másolja. Ez például a /var/log könyvtár esetében jól jöhet, mert végig tudjuk nézni rendszergazdaként, hogy melyik klienst mikor és mennyit használták.

A /etc könyvtár alaposabb átböngészést igényel. Ilyenkor semmi esetre se tegyünk a /etc-ből olyan file-okat a basedirbe, mint a sysconfig/network, sysconfig/network-scripts/ifcfg-eth0, hostname, HOSTNAME, mtab, fstab, init.d/network stb. A /etc könyvtár nagy része szintén linkelhető az összes, vagy az azonos típusú (megegyező x,y) kliensek között. De ha a fent felsorolt állományokat a linkeltek közé vesszük, akkor nagy galibát okozhatunk, és eltarthat egy darabig annak megtalálása, hogy miért nem úgy működik a kliens, mint ahogy azt vártuk. Az összes file-t és könyvtárat egy előre megadott könyvtárba kell tenni, ami a példában a /X. A program nemcsak innen veszi a kiinduláshoz (másoláshoz/linkeléshez) szükséges file-okat, hanem ide is teszi az IP-számnak megfelelően.

Miután beállítottuk a klienst, egy kicsit foglalkozzunk az alkalmazásszerverünkkel. Azt a gépet, amit kijelölünk arra a feladatra, hogy a nagyobb alkalmazásokat (például Netscape, StarOffice, Applix stb.) futtassuk rajta, szintén konfigurálni kell. A gépen állandóan futnia kell az xdm, vagy kdm démonok valamelyikének. Ezek biztosítják a bejelentkezési lehetőséget az X-terminálok grafikus felületéről, valamint az ablakkezelőt (WindowManager) is ezek indítják. Ha a szerverünkön is állandóan fut valamelyik virtuális konzolon az xdm bejelentkezési ablaka, akkor nyilván fel is vannak telepítve a szükséges programok. Ha eddig nem telepítettük a grafikus felületet és az X11-et használó programokat, most tegyük ezt meg. Utána egy egyszerű konfigurálási feladatunk marad csak: ha azt szeretnénk, hogy az alkalmazásszerveren is fusson valamelyik VT-n grafikus felület, akkor hagyjuk meg az erre vonatkozó sort a /etc/X11/xdm/Xservers-ben. Ha az alkalmazásszerver konzolját csak szövegesen használjuk, akkor pedig kommentezzük ki. (Ne töröljük ki, hátha kell még!)

Most jön a kliensek beírása ebbe a file-ba. Minden sorba bejegyzünk egy klienst, úgy hogy beírjuk a nevét (ha neve nincs, az IP-számát), és egy kettősponttal elválasztva az adott képernyő sorszámát. (Nem valószínű, hogy egy kliensről több virtuális konzolon is szeretnénk használni grafikus felületet, de akár több is megadható külön sorokban. Ne felejtsük el, hogy a sorszámozás a 0-tól kezdődik!) Ezután írjuk be, hogy távoli gépekről van szó. Íme a példa, hogy a fenti gep1,gep2 nevű gépekhez mit kell bejegyezni (a 10.0.0.0-ás tartományt nem érdemes a DNS-be bejegyezni, elég ha a /etc/hostsban szerepel a gép, és az IP-cím).

```
gep1:0 foreign
gep2:0 foreign
```

Ha bejegyeztük, indítsuk újra az xdm-et! Próbáljuk ki valamelyik kliensen, hogy működik-e a dolog:

```
[root@gep1 /]# /etc/X11/X -query szerver &
```

Ha bejön egy bejelentkeztető ablak, és be is tudunk jelentkezni, akkor minden rendben ment, és van egy többékevésbé teljes értékű X-terminálunk.

De ez még így nem indul automatikusan. Disztribúciója válogatja, hogy az xdm-et, hogy indítják. Most az X-terminálunkon nem kell, hogy az xdm fusson, hiszen az az alkal-

mazáskiszolgálón fut. Nekünk csak a fent említett sort kell lefuttatni minden egyes indításkor. A cél eléréséhez módosítani kell a kliensek /etc/inittabját azzal, hogy az alábbi sort hozzátesszük:

x:2345:respawn:/etc/X11/X -query szerver

Persze az /etc/inittab módosításával tovább finomíthatjuk egy X-terminál kinézetét, például beállíthatjuk, hogy egy helyi szöveges konzolon korlátozott lehetőségekkel be lehessen jelentkezni vagy egy szöveges módú böngészőprogramot is tehetünk valamelyik virtuális terminálra. Ez hasznos, ha valakinek nincs szüksége a grafikus felületre, csak gyors telnet programot akar levélolvasáshoz, vagy ftp-letöltéshez: esetleg egy kis fogyasztású, de gyors böngészőprogramot, mint amilyen a lynx.

További ötletek:

● Az X-szerver kimeneteinek és hibaüzeneteinek naplózása (egy sor): x:2345:respawn:/etc/X11/X -query >/var/log/Xstdout 2>/var/log/Xstderr

● Egy lynx böngésző futtatása valamelyik VT-n (A példában az 5-ösön): 5:2345:respawn:path-to-start-lynx http://nyitolap URL /dev/tty5

Itt figyeljünk arra, hogy ne közvetlenül a lynxet indítsuk, hanem valamilyen anonym felhasználó nevében egy kis shellszkriptből futtassuk. Ugyanis, ha a root jogosultságokkal elindítunk egy lynxet, akkor a rendszert – de legalábbis a klienseket – nagyon gyorsan fel lehet törni! Gondoljunk csak arra, hogy valaki egy weboldalról letölt egy lebutított passwd file-t, amiben csak egy jelszó nélküli root felhasználó van, és ezt elmenti a megfelelő helyre.

● A bejelentkezési lehetőségeket még finomabban is szabályozhatjuk. Megadhatjuk, hogy hány és mely VT-ken engedélyezzük a bejelentkezést a getty, mingetty és hasonló programok futtatásával.

Hogy a példákhoz némi magyarázattal is szolgáljak, az inittab szerkezete a következő:

id:runl:action:command

● id: Minden egyes sort külön azonosítóval kell ellátnunk. A szöveges bejelentkeztető programok igénylik, hogy ez az azonosító megegyezzen a virtuális terminál sorszámával.

● runl: Azon runlevelek felsorolása, amelyben szeretnénk, ha a fenti sor érvényesülne. Nem minden esetben kell kitölteni.

● action: Milyen esemény bekövetkeztekor kell végrehajtani a sort, és hogyan. Ilyenek például a sysinit, amely a rendszerindításkor hajtja végre a sort (ez az eset, amikor nem kell kitölteni a runl mezőt) vagy a respawn, amely megpróbálja végrehajtani a parancsot. És ha befejeződött, akkor újra elindítja, és újra és újra...

● command: Az utasítás, amit az adott esetben végre kell hajtani. Korábban szerepelt, hogy esetleg belső/privát hálózaton kell beállítani az X-terminálokat, mégpedig oly módon, hogy sok az X-terminál, de kevés, vagy csak egy IP-címünk van. Az egyetlen cím az Interneten lévő gépé, azaz a gateway-é (hálózati átjáró). Nem nehéz feladat úgy beállítani a hálózatot, hogy habár az X-terminálok fizikailag vagy logikailag (az ő 10.x.y.z,etc. IP-számaikat nem továbbítják a hálózati átjárók/gateway-ek) el vannak szigetelve az Internettől, de mégis láthassák azt. Hiszen mi értelme volna például egy lynxet futtatni egy szöveges konzolon, ha nincs is mit böngészni? Erre a választ egy NAT nevű technika speciális ága jelenti.

NAT, de nem Nemzeti Alaptanterv

A NAT (Network Address Translation – hálózati címfordítás) egy olyan technika, amelyet a hálózati átjárókra találtak ki a célból, hogy ha van az Internet felé is egy hálózati kapcsolat, és van egy belső hálózat, akkor a kettő közt a címet fordítsa. A címfordítás menete a következő: ha például az egyik – 10.x.y.z belső című – gépünk lekér valamit az Internetről, akkor azt hiszi a hívott gép, hogy egy létező internetes címről keresték. Ezen az álcímen el is lehet érni a belső gépet az Internet oldaláról, és ez az álcím különbözik az átjáró címétől. A rossz hír az, hogy ezt a technikát még hivatalosan a v2.0 kernelek nem támogatták, csak nem-hivatalos kernelpatcheket lehetett letölteni a "linux mamáról". A kérdés az, hogy akkor miért jó nekünk egy ilyen, még nem is támogatott technika? A válasz egyszerű. Azért, mert ennek a technikának van egy lebutított változata is, amely a belső címeket mind a sajátjára fordítja. Így viszont elveszítjük azt a lehetőséget, hogy a belső gépeinket kívülről is el lehessen érni. Persze, ha megnyílik egy adatcsatorna, az kétirányú lesz, mindaddig míg be nem zárul, így például ha kiküldünk egy HTTP kérést, akkor ameddig be nem zárjuk az adatcsatornát, a válasz is visszajön, tehát megérkezik a kért weboldal.

Ennek a technikának a neve: IP-Masquerading. Ezt viszont teljesen támogatják már a v2.0-s Linux kernelek is. Ilyenkor az átjáró kernelébe bele kell fordítani a hálózati támogatást, a TCP/IP hálózati támogatást, az átjárók támogatását, valamint az IP-Masqueradinget. Ha v2.0-s kernelünk van, akkor az ipfwadm nevű programmal beállíthatjuk az IP-Masqueradinget is. Ha v2.1-es vagy v2.2-es kernelünk van, akkor már az ipchainst kell használnunk. Példaként álljon itt az a parancs, amit ahhoz kell kiadnunk, hogy a 10.x.y.z hálózatot maszkolja a gateway az Internet felé:

[root@gateway /]# ipfwadm -F -a m -S 10.0.0.0/8

(Az IP-maszkolásról, az ipfwadm és az ipchains használatáról a CHIP Magazinban már többször is volt szó, ajánljuk olvasásra. – a szerk.)

Persze, ez a parancs csak addig marad érvényben, amíg le nem állítjuk az átjárógépet, hiszen a Linux csak a memóriában tárolja azt, hogy milyen szabályok érvényesek rá, mint átjáróra. Ha azt szeretnénk, hogy ez a gép minden indításakor beállítódjon, akkor tegyük egy olyan shellszkriptbe, amiben biztosan le fog futni minden rendszerindításkor a megfelelő időben: lehetőleg a hálózat beállítása után. Azért vigyázzunk arra, hogy az olyan disztribúciókban, mint a Debian, vannak már ipfwadm parancsok bizonyos szűrések elvégzésére, a hálózati betolakodók kívül tartása érdekében. Persze ebben a disztribúcióban létezik olyan csomag, ami elvégzi helyettünk ezek beállítását, de szükség lehet arra, hogy itt-ott enyhítsünk vagy szigorítsunk a szabályokon. Mindenféleképp szánjunk néhány percet arra, hogy átnézzük ezeket a szkripteket. Arra is érdemes időt szentelni, hogy az ipchainst tanulmányozzuk, hiszen a későbbi kernelverziók esetén már ez lesz a konfiguráló program, amelyet használnunk kell.

Azzal, hogy az IP-Masquerading minden gép címét az átjáróéra fordítja, felvet egy újabb problémát. Mi történik akkor, ha valakinek nemcsak a helyi alkalmazásszerveren van lehetősége alkalmazásokat futtatni, hanem egy olyan helyen is, amely kívül esik a belső hálózaton? Mielőtt ezt a kérdést megválaszolnám teszek egy kis kitérőt. Lássuk, hogy mi a helyzet több szerver esetén.

Az X-rablók támadása

Aki megpróbált esetleg más szerverről – vagy akár ugyanarra a gépre betelnetelve és más felhasználói azonosítót használva – programokat (ne feledjük, ezek az X-kliensek) indítani, az szomorúan tapasztalhatta, hogy vissza lett utasítva kérése azzal, hogy nem tud kapcsolódni az X-kliense az X-szerverhez. Ha nem mi hibáztunk, és nem is a rendszer, akkor a beállításokban kell keresni a hibát. Nyilván az alapértelmezés miatt van az, hogy csak azokat az ügyfeleket engedik be az X-szerverek, amelyek már egyszer azonosítva lettek. Ezt az azonosítást az xdm-től nem követelte meg az X-szerver, hiszen eleve tőle kérte a bejelentkeztetést a program (ez volt a -query), és így az sem ütközött akadályokba, hogy egy ablakkezelőt (Window Managert), esetleg néhány grafikus ablakot indítson. De vajon mi újság a többiekkel?

Erre valók az xauth és xhost parancsok. Velük lehet szabályozni, hogy kik és milyen protokollok szerint kapcsolódhatnak az X-szerverünkhöz. Az xauth-tal finoman és egyénre szabottan beállíthatja mindenki magának, hogy mit kíván engedélyezni. Az általánosan elterjedt viszont az (a nem teljesen biztonságos megoldás), hogy mindenkinek megengedjük, hogy kapcsolódhasson az X-szerverünkhöz.

Ezt egyrészt tekinthetjük biztonsági lyuknak is, mert boszszantó, ha valaki küld a képernyőmre egy akkora méretű Netscape ablakot, hogy csak a közepe látszik, és sehogy sem tudom bezárni. De tekinthetjük külön szolgáltatásnak is, hogy bárki küldhet a képernyőnkre például egy üzenetet (már ha tudja az X-terminálom hálózati címét), amiben megérdeklődi, hogy ráérek-e egy kis szóbeli csevegésre, és akár még interaktív válaszadási lehetőséget is felkínál. Nézőpont kérdése az egész. Persze nem valószínű, hogy bárki tudja a szomszéd teremből, hogy melyik gépnél ülök (ha csak nem ismeri a finger parancsot), de az sem valószínű, hogy egy betörő pont az én gépemre pályázik. Igaz, az ilyen „Miért pont hozzám törnének be?" kérdést feltevő rendszergazdákhoz szoktak leggyakrabban betörni.

Állapodjunk meg abban, hogy a rendszer automatikusan beállítja, hogy bárki kapcsolódhasson, és ezt a felhasználó finomíthatja igényei szerint, megfelelő autorizációs file létrehozásával, valamint a bejelentkeztetés utáni xauth megfelelő lefuttatásával. A gyakorlat azt mutatja, hogy nem szoktak eltérni ettől a beállítástól. (Igen jó, igaz nem teljesen ingyenes megoldás az ssh használata, mely egy kódolt csatornán továbbítani tudja az X protokollt is. Ekkor nincs szükség a fenti módszerre. – a szerk.)

Tehát ahhoz, hogy bárki automatikusan kapcsolódhasson, csak annyi a dolgunk, hogy a bejelentkezés után automatikusan lefusson az „xhost +" parancs. Ezt a hatást elérhetjük oly módon is, hogy betesszük a parancsot az xdm-et futtató gépen az Xsession, Xclients nevű file-ok valamelyikébe. Lehetőleg az elejére írjuk (Vigyázat: ne az első sorba!) és ne olyan részre tegyük, ami esetleg le sem fut, vagy csak olyankor, amikor már épp kijelentkezett a felhasználó! Ugyanennek a lehetőségnek a „hangos" párja, hogy bármelyik gépről megszólaltathassák a hangszórónkat. Ennek a beállítása sem igényel sokkal több munkát, csak épp nem az alkalmazásszerveren kell bütykölni, hanem az AU-szerveren, közvetlenül. A dolgunk mindössze annyi, hogy megkeressük azt a részt a kliensek initszkriptjei közt, amelyik elindítja a NAS-t. Ez gyakorlatilag annyit tesz, hogy „démonizálva" elindítja a /usr/X11R6/bin/au-t. Ezen csak annyit kell változtatni, hogy utána írjuk még a -aa kapcsolót a parancssorba.

Hibátlan álarc

Most e rövid kitérő után térjünk vissza arra, hogy hogyan küszöböljük ki az IP-Masquerading által okozott problémát.

Az X-szerverekről tudni kell még azt is, hogy egy gépen nemcsak egy futhat, hanem több is, különböző displayeken, amelyek mind el vannak látva egy sorszámmal. Természetesen nullával kezdődik a számozás. Ha már valaki kipróbálta az eddig beállított X-terminálját, és lekérte egy az ablakkezelő segítségével indított parancssor környezeti vál-

tozóit (env vagy set parancs), akkor tapasztalhatta, hogy van is egy ilyen DISPLAY nevű változó. Ebben pedig értékként annak a DISPLAY-nek a címe, amelyiken dolgozunk, gépnév vagy „IPszám:display-száma.képernyőszáma" formátumban.

(A localhost hostnév és a 0 mint képernyőszám elhagyható. – a szerk.) Ebben csak a képernyő száma az új elem, ami nem is mindig szerepel, de számunkra ez most nem is lényeges. A trükk ott rejtőzik, hogy úgy tesz a rendszer (ha beállítottuk), mintha a gateway-en futna a rengeteg X-szerver. Ehhez csak annyit kell beállítani, hogy amit a rendszer az átjáró 1-es display-ére küld, azt az első gép 0-ás display-ére továbbítsa, amit a 2-esre azt a második gép 0-ás display-ére, és így tovább. Ennek beállítására is több módszer létezik.

● Az átjáró kernelébe belefordítjuk az IP Auto Forwarding opciót. Letöltjük hozzá az ipautofw nevű programot, és beállítjuk, hogy minden rendszerindításkor lefusson az a parancssorozat, ami ezeket a továbbításokat elvégzi. Tapasztalataim szerint ez a leglassabb mód, még akkor is ha kernelből van támogatva.

● Beszerezzük/feltelepítjük a redir nevű programot, és ezzel oldjuk meg a továbbítást. Azt, hogy ezzel hogyan tesszük meg arra is több lehetőség kínálkozik.

– Megtehetjük úgy, hogy a rendszerindításkor annyiszor indítjuk el a redirt, ahány gépünk van a belső hálózaton.

– De úgy is, hogy a /etc/services,/etc/inetd.conf file-okba beírjuk a megfelelő sorokat.

– Két példányt is indíthatunk automatikusan az inetd-ből különböző konfigurációs állományokkal, és a másodikban csak a továbbításért felelős bejegyzések vannak, míg az elsőt meghagyjuk eredetiben. (A /etc/servicesbe ettől függetlenül ugyanúgy be kell őket írni.)

(A fentinél sokkal elegánsabb megoldás az xproxy használata, mely pont ennek a problémának a leküzdését szolgálja, és szintén része a Debian Linuxnak. – a szerk.)

Válasszunk ízlésünk szerint a fentiek közül, de az utolsó kettő jár a legkevesebb fáradsággal és a legjobb teljesítménnyel. Már csak azt nem tudjuk, hogy pontosan mit kellene, és hova továbbítani. Az X-display-ek a TCP/IP-ben ugyanúgy kiérdemeltek egy portot, mint a HTTP vagy az FTP. Az övék: 6000+offset, amit úgy kell érteni, hogy a 6000-es port a 0-ás display-é, a 6001-es az 1-es display-é és így tovább. Tehát az átjáró 6001-es portjára érkező adatokat továbbítani kell a belső hálózaton lévő első gép 6000-es portjára, az átjáró 6002-es portjára érkező adatokat a második gép 6000-es portjára, és így tovább. A hangokkal ugyanígy megy a dolog, csak ott a 8000-hez kell hozzáadni a display sorszámát.

Ez utóbbi két megoldás szerint a /etc/services-be az 5. listán látható sorokat, míg az inetd.conf-ba a 6. listán látható sorokat kell beírni.

```
Disp1 6001/tcp # X11 gep1 részére
Disp2 6002/tcp # X11 gep2 részére
...
Audio1 8001/tcp # Audio gep1 részére
Audio2 8002/tcp # Audio gep2 részére
...
Példa a /etc/services-be beírandó sorokra, hogy
/etc/inetd.conf-ból indíthassuk a redirt.

Disp1 stream tcp nowait root /usr/sbin/tcpd redir –inetd
10.0.1.1 6000
Disp2 stream tcp nowait root /usr/sbin/tcpd redir –inetd
10.0.1.2 6000
...
Audio1 stream tcp nowait root /usr/sbin/tcpd redir –inetd
10.0.1.1 8000
Audio2 stream tcp nowait root /usr/sbin/tcpd redir –inetd
10.0.1.2 8000
...
# Részletek ahhoz, hogy mit írjunk be a /etc/inetd.conf-ba
# ahhoz, hogy az automatikusan elindítsa a redir-t.
```

Most, hogy ilyen szépen megoldottuk a problémát, okoztunk egy másikat, de ez már lényegesen kisebb, ráadásul teljesen kiküszöbölhető. A probléma csupán annyi, hogy ha egy külső gépről akarunk programot indítani, akkor mindig újra be kell állítani a DISPLAY értékét, kézzel. Ez szintén egy olyan dolog, amit könnyen automatizálni lehet. De előbb készítsünk egy táblázatot, amibe minden sorba beírjuk, hogy mire kell változtatni a DISPLAY értékét, és hogy miről. Egy példa:

gateway.valahol.hu:1 gep1:0.0
gateway.valahol.hu:2 gep2:0.0

Ezt a táblázatot mentsük el /etc/X11/displays néven. Már csak azt a részt kell beleírni az Xsession illetve az Xclients valamelyikébe, amely elvégzi ezt a cserét. A beszúrandó példát a 7. listában találjuk

```
/usr/X11R6/bin/xhost + 2>&1 >>/dev/null
x=`cat /etc/X11/displays|grep „<DISPLAY"|awk '{ print <1 }'`
DISPLAY=<{x:-"<DISPLAY"}
export DISPLAY
# Ezzel a részlettel kell kiegészíteni az Xsession-t, ha azt
# akarjuk, hogy a DISPLAY értékét automatikusan beállítsa
```

Ha ezt beszúrtuk, akkor már minden olyan programban, amit az Xsession/Xclients indított, már ez az érték fog szerepelni, hiszen az export utasítás miatt az öröklődik. Tehát ha az ablakkezelő segítségével például a root menüből vagy a start menüből elindítunk egy parancssort, amiben kiadjuk a telnet utasítást, a DISPLAY értéke is továbbítódni fog a későbbiekben indított utasításoknak. Az X-termináljaink ezek után mindent tudnak, amit egy „jólnevelt" X-terminálnak tudnia kell. Előfordulhat, hogy a rendszerben nincs mindenkinek felhasználói azonosítója vagy olyanoknak is szeretnénk internetezési lehetőséget biztosítani, akiknek nincs.

Netscape-nevelés

Ugyebár a laikus felhasználó weböngészésre a Netscape-et használja. Ha létrehozunk egy felhasználót, amihez

nincs jelszó, az nem fog tudni több példányt elindítani a Netscape-ből, mert az összezavarná az ideiglenesen letöltött file-jait (cache-ét), és rögtön hibaüzenettel indulna, ami a tapasztalatlan felhasználót elbizonytalanítja. De szerencsére, itt is alkalmazható egy kis trükk. Ez annyit takar, hogy minden X-terminálnál más és más home-könyvtárat lát a Netscape.

A dolgunk csupán annyi marad, hogy létrehozzunk egy felhasználói azonosítót, amihez nincs jelszó (legyen a neve a példánkban kliens), belépünk rajta egyszer, és elindítjuk a Netscape-et. Ha ezzel megvagyunk, akkor a felhasználó home-könyvtárát bemásoljuk a /X/basedirc/home/kliens könyvtárba. Ezek után, amikor az általunk írt mkclient paranccsal létrehozzuk az adott géphez tartozó file-rendszert, akkor minden egyes géphez létrejön egy home-könyvtár, amiben a kliens eredeti file-jairól van egy másolat. (Fontos, hogy nem link! Ha szükséges, vegyük a root tulajdonába a Netscape konfigurációs állományát, hogy a felhasználó semmilyen módon ne tudja nagyobbra venni a cache méretét!)

Eddig ott tartunk, hogy minden egyes kliensgéphez létrejön egy-egy külön home-könyvtár. Tegyünk róla, hogy ha valaki ezzel az azonosítóval lép be, akkor mindig megváltozzon a HOME változó értéke! Ez megtörténhetne a /X/10.x.y.z/home/kliens-re is, de szebb, ha a /home/kliens könyvtárban szimbolikus linkeket hozunk létre az adott kliensek neveivel és ezek mutatnak a /X/10.x.y.z/home/kliensre.

```
[/home/kliens]# ln /X/10.0.1.1/home/kliens gep1 -s
[/home/kliens]# ln /X/10.0.1.2/home/kliens gep2 -s
```

Megjegyzés: a példámban a szimbolikus linkek nevei az X-terminálok esetén előforduló DISPLAY környezeti változóban előforduló gépnevek. Most már csak azt kell megtennünk, hogy az alkalmazásszerveren az Xsession vagy Xclients file-t, ahol megspékeltük az előbb, a 8. listán látható kis módosítással toldjuk meg.

```
/usr/X11R6/bin/xhost + 2>&1 >>/dev/null
if [ <HOME = „/home/kliens" ]; then
 HOME=/home/kliens/`echo „<DISPLAY" |awk -F: ' { print <1 }
'|awk -F. ' { print <1 }'`
fi;
x=`cat /etc/X11/xdm/terms|grep „<DISPLAY"|awk '{ print <1 }'`
DISPLAY=<{x:-"<DISPLAY"}
export DISPLAY
# Ezzel a részlettel kell kiegészíteni az Xsession-t, ha azt
# akarjuk, hogy a DISPLAY és a HOME értékeit automatikusan
# beállítsa
```

Ezután, ha a kliensfelhasználó bejelentkezik, akkor minden egyes X-terminálról el tud egy példányt indítani a Netscape-ből. Ezek után már tökéletesek az X-termináljaink – és rengeteg lehetőség rejtőzik még előlünk. Például a fenti mkclient szkriptben, ahol az fstabot már eleve úgy állította be, hogy minden kliensen a hozzá tartozó home-ot mountolta fel a kliens felhasználó homekönyvtáraként.

Perifériális kérdések

Lehetnek olyan felhasználóink is, akik szeretnének hozzáférni az X-terminálok futtatása közben is a PC-k CD-ROM-jaihoz vagy merevlemezéhez (már ha van), esetleg floppyjaihoz. A PC-k merevlemezével nincs különösebb gond, mert azt a kliensek a bootoláskor felmountolhatják, és NFS-en exportálhatják a szervernek. A helyzet hasonló a CD-ROM-ok és floppymeghajtók esetében is. Ilyenkor azonban már picit összetettebb a helyzet. Ezek a kliensre felmountolhatók az auto-fs segítségével is, de az adott gépről bejelentkezett felhasználó is felmountoltathatja őket. Jó ötlet, hogy létrehozunk a klienseken egy olyan felhasználói azonosítót, ami lockolva van (jelszóval nem lehet belépni). Ha feltelepítettük a sudo-t a kliensre, akkor megadjuk, hogy ez a felhasználó mountolhassa a floppyt, vagy a CD-ROM-ot (le vagy föl). Biztonsági okokból célszerű azonban megadni, hogy csak a nosuid opcióval együtt tehesse ezt meg. Ha ezt nem adjuk meg, akkor valaki otthon készít egy lemezt ext2-fsel, vagy umsdos-zal, amin csak egy egyszerű shellt indító parancsszkript van, erre beállítja a setuid bitet, és már kész is a betörés. Persze nem az a célom, hogy betörési tippeket adjak. Ennek a felhasználónak a home-könyvtárába tegyünk egy .rhosts nevű file-t, amiben engedélyezzük, hogy a szerverről a rootfelhasználó átjelentkezhessen az rlogin, rexec, rsh stb. parancsokkal. A szerveren pedig az Xsession, Xclients parancsokból adminisztráljuk a ki- és bejelentkezéseket, és nyomon követjük, hogy melyik X-terminálról ki van bejelentkezve. Ezen túlmenően az ablakkezelő (WM) menüjébe is betehetjük, hogy a szerveren elindítson egy általunk írt setuides programot, ami elindítja az rsh -l username parancs paraméterek formátumú paranccsal meghívva a kliensen a felmountolást. Ezután helyben is elvégzi a felmountolást az NFS-en keresztül, és így a szerverről látszanak az X-terminálok erőforrásai. (Akár a felhasználó home-könyvtárába is tehetünk ideiglenesen egy szimbolikus linket arra a könyvtárra, ahová felmountoltuk az X-terminál erőforrását.) Persze a setuides programot a szerveren úgy írjuk meg, hogy ellenőrizze, tényleg az adott felhasználó van-e bejelentkezve az adott gépről, nem pedig egy kívülálló próbálkozik. Ennyi shellszkript megírása után már elég gyakorlatunk lesz ahhoz, hogy követni tudjuk az X-terminálokon való be- és kijelentkezéseket, valamint ellenőrizni tudjuk azokat. Egy primitív, de hasznos ötlet például, ha egy speciálisan erre a célra készült könyvtárban létrehozunk minden klienshez egy üres file-t, és ebbe a bejelentkezéskor beírjuk, hogy ki jelentkezett be arról a gépről, és a kijelentkezésekor pedig kiürítjük ezt a file-t.

Remélem, sikerült sok hasznos tippet és segítséget adnom a PC-k X-terminálosítására. Amennyiben valakinek további kérdése lenne az e-mail címem a cikk elején megtalálható. ■

Nagy Balázs
julian7@kva.hu

Levelezés Qmail segítségével

A Qmail egy biztonságos, megbízható, hatékony és végül, de nem utolsó sorban egyszerű levéltovábbító rendszer (MTA), amivel kiváltható a legtöbb Unix rendszeren alapértelmezett, teljes sendmail-binmail rendszer.

Biztonságos: a biztonság nem csupán cél, hanem minimális követelmény. A levéltovábbítás a felhasználók számára kritikus; nem lehet kikapcsolni, tehát mindenképpen biztonságosnak kell lennie. Megbízható: a Qmail „egyenes út" filozófiája garantálja, hogy ha megérkezik egy levél, az nem fog elveszni. A Qmail ráadásul bevezetett egy teljesen megbízható levelesláda formátumot, amivel nem csak közzé lehet tenni a levelesládát NFS-en, de bármennyi NFS-kliens továbbíthat ugyanabba a levelesládába egy időben. Hatékony: egy Pentiumon futó BSD rendszeren a Qmail könnyedén feldolgoz naponta akár 200 ezer különálló helyi levelet. Mivel az SMTP-kapcsolat létrehozása lassú, ezért a Qmail egyszerre húsz vagy több küldést is végezhet, ami drasztikusan csökkenti például a levelezőlisták levélküldéseinek időigényét. Egyszerű: a Qmail bármelyik más levéltovábbító rendszernél kisebb, többek között azért, mert a többi MTA különválasztotta a továbbküldést, a más névre továbbítást és a levelezőlista-mechanizmusokat, míg a Qmail ezeket egyesítette. Kiváltható a sendmail: a Qmail számos eljárást (pl.: virtuális gépek) támogat, ami a modern MTA-k sajátja. Ráadásul tartalmaz egy /usr/lib/sendmail klónt is, így a levélküldés megváltozása nem okoz semmilyen problémát.

A telepítés előkészítése

Ha belevágunk, akkor készüljünk fel rá, hogy a teljes levelezést megbolygatjuk. Így valószínűleg használhatatlan lesz az eredeti levelesládánk, a levelezőprogramunk (ugyanezen okból), a POP3 kiszolgálónk, a procmailünk, hogy a finger és a login parancsok levélfigyelő részeit már ne is említsük. Legelőször el kell dönteni, hogy milyen levelesládát használunk. Erre két lehetőségünk is nyílik.

1. Minden felhasználó a saját könyvtárába kapja a leveleit, a ~/Mailbox levelesládába. Ilyenkor megvan a lehetőség arra, hogy a régi helyre átláncoljuk a levelesládát, így a régi programokat nem is kell lecserélni.

2. Átváltunk maildirre. Ekkor igaz, hogy ki kell cserélni a levélolvasó-klienseket, de cserébe használhatjuk a Qmail eredeti POP3 kiszolgálóját.

Ezután nekiállhatunk lefordítani a qmail programot. Ehhez az alábbiakat kell tennünk:

```
tar zxvf qmail-1.03.tar.gz
mkdir /var/qmail
groupadd nofiles
useradd -g nofiles -d /var/qmail/alias alias
useradd -g nofiles -d /var/qmail qmaild
useradd -g nofiles -d /var/qmail qmaill
useradd -g nofiles -d /var/qmail qmailp
groupadd qmail
useradd -g qmail -d /var/qmail qmailq
useradd -g qmail -d /var/qmail qmailr
useradd -g qmail -d /var/qmail qmails
make setup check
./config
```

Ha nincs groupadd és useradd a gépen, akkor a /etc/passwd és /etc/group file-okba írjuk a táblázatban látható sorokat.

/etc/passwd:

```
alias:*:7790:2108::/var/qmail/alias:/bin/true
qmaild:*:7791:2108::/var/qmail:/bin/true
qmaill:*:7792:2108::/var/qmail:/bin/true
qmailp:*:7793:2108::/var/qmail:/bin/true
qmailq:*:7794:2107::/var/qmail:/bin/true
qmailr:*:7795:2107::/var/qmail:/bin/true
qmails:*:7796:2107::/var/qmail:/bin/true
```

/etc/group:

```
qmail:*:2107:
nofiles:*:2108:
```

Természetesen a 2107 és 2108-as számú GID-ek és a 77907796 UID-ek helyett bármilyen, még nem használt számok lehetnek, de az összefüggéseket tartsuk meg. Hiányosan telepített rendszer esetén (pl.: nem talál DNS-kiszolgálót a gép adataihoz) a ./config nem működik, helyette a './config-fast <gépnév>' parancsot kell kiadni. Most jutottunk el addig a pontig, ameddig automatikusan el lehet jutni. Ezután következnek még a „finomságok".

Beállítások

A Qmail rendszer összes file-ja a /var/qmail könyvtárban található, így a relatív útvonalak ehhez a könyvtárhoz tartoznak a rendszerben. Tehát a beállítható adatok a control/ alkönyvtárban vannak. Ezek közül a legfontosabb a 'me', ami a gazdagép nevét tartalmazza. Általánosságban elmondható a konfigurációs file-okról, hogy két csoportba oszthatók aszerint, hogy egy adatot vagy egy egész listát tartalmaznak-e. Ez utóbbi minden eleme új sorba kerül. A teljesség igénye nélkül álljanak itt a fontosabb file-ok leírásai.

defaultdomain:
A relatív e-mail címekbe (amikben nincs @) behelyettesítendő gép neve.

locals:
Azon gépnevek, amelyekre az érkező leveleket helyiként kell értelmezni. Ide bármit beírhatunk, tehát ha például egy belső Windows-hálózaton a bela@mazsola.tade.com címre írnak levelet, a Windows úgy tudja, hogy a mazsola.tade.com a Qmail kiszolgáló, és ha a localsban szerepel is ez a név, akkor a levelet helyben kézbesíti a qmail-local.

rcpthosts:
Ide lehet beírni, hogy milyen gépekre engedélyezett a levélküldés. Ez igazából arra jó, hogy az Internetről ne használják gépünket levelező-átjáróként (relay). (Ezt meg lehet kerülni, de erről majd később.)

badmailfrom:
Ebbe a file-ba lehet beírni, hogy kiktől nem fogadunk el levelet (a From: mező alapján). Ha '@gépnév' formában írjuk, akkor arról a gépről egyáltalán nem fogadunk el levelet.

concurrencylocal:
Ide lehet írni azt a számot, ahány helyi továbbítást engedélyezünk egyszerre. Trükkös használatával el lehet érni azt is, hogy a helyi továbbítások mennek, de a távoliak le vannak tiltva (pl.: tudjuk, hogy nincs fent a gép a hálón). 0 a letiltást jelenti.

concurrencyremote:
Itt lehet megadni, hogy hány távoli levelet továbbítson egyszerre a gép. Az alapbeállítás 20, de a krondor.kva.hu (amely egy 32 MB-os Pentium 200) elég jól viseli a százat is, és – szerintem – gyorsabban küldi el a listák leveleit. Természetesen a 0 itt is a letiltást jelenti.

databytes:
Ez a file tartalmazhatja a beérkező levelek maximális hosszát. A 0 itt a korlátozás feloldását jelenti.

virtualdomains:
Ide lehet beírni a virtuális gépek (felhasználók) nevét, és az ahhoz tartozó alias előtagját, egymástól kettősponttal elválasztva. Például a ceg.hu:ceg sor azt jelenti, hogy az összes ceg.hu-ra érkező levelet a ceg nevű helyi felhasználó fogja megkapni valaki@ceg.hu → ceg-valaki@ceg.hu átalakítással.

smtproutes:
Ide lehet beírni azon gépek nevét és levéltovábbítási útvonalát, amiknek külön meghatározzuk az útját. Így lehetséges a leveleket tűzfalon átküldeni, de már sikerült így összekötnöm egy MUD-ot az Internettel levelezés szintjén.

Ez is kettősponttal elválasztott lista, aminek az első oszlopa a gép neve, a második azon gép neve, ahova tovább kell küldeni a levelet, a harmadik oszlopban pedig meg lehet adni a portszámot (már amennyiben nem a szabványos 25-ös (SMTP) portra kell küldeni a levelet).

Például az After the Plague MUD (Multi User Dungeon, avagy többfelhasználós kalandjáték) leveleit így irányítottuk át: atp.szerver.neve:host.szerver.neve:3003, ami az összes atp.szerver.neve címre küldendő levelet a host.szerver.neve gép 3003-as portjára küldi SMTP protokollal. A /var/qmail/alias könyvtárban (és minden felhasználó saját könyvtárában) a .qmaillel kezdődő file-ok határozzák meg a beérkező levelek sorsát. A fő /aliasok a var/qmail/alias könyvtárban vannak, míg a felhasználó a saját könyvtárában levő .qmail file-okkal állíthatja be leveleinek útvonalát. Ezekben a file-okban soronként egy utasítás lehet, amit a levélküldés elvégez. A sorok első eleme az utasítás, a többi az utasítás paramétere. Az utasítások a következő táblázatban találhatók.

#	megjegyzés
| prog	program indítás
./fajl	MBOX levelesláda a saját könyvtártól (nincs a végén /)
/konyvtar/fajl	MBOX levelesláda (nincs a végén /)
./mdir/	Maildir levelesláda a saját könyvtártól (/ van a végén)
/konyvt/mdir/	Maildir levelesláda (/ van a végén)
& <email cím>	továbbítás (ha nem #, |, / vagy . van az e-mail cím elején, akkor nem szükséges a & jel)

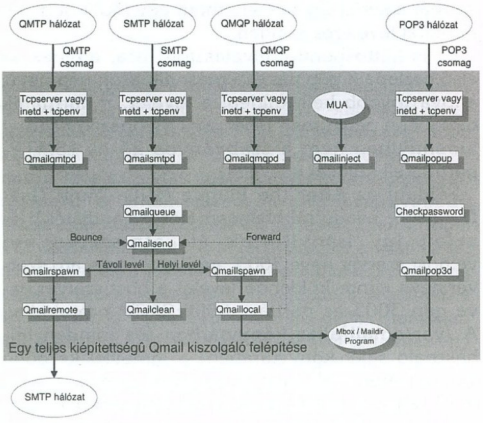
A levelesládába továbbítás kezdődhet ./ jellel is, amivel a saját könyvtáron belüli relatív útvonalat adhatjuk meg.

Ismerjük meg a Qmail működését!

Mielőtt elindítjuk a Qmailt, ismernünk kell azt, hogy miként dolgozik. Nagy vonalakban a Qmail három tevékenységet végez:

1. fogadja a beérkező leveleket,
2. ezeket feldolgozza és tárolja,
3. majd elküldi a címzetthez.

A program működését az alábbi ábrán követhetjük nyomon.



Egy teljes kiépítettségű Qmail kiszolgáló felépítése

Az első tevékenység négyféleképpen történhet: helyi programokkal (qmail-inject, sendmail), QMTP (209/tcp), QMQP (628/tcp), vagy SMTP (25/tcp) porton keresztül. Ez utóbbiakról annyit, hogy az SMTP a napjainkban leggyakrabban használt levélküldési módszer, míg a QMTP-t (Quick Mail Transfer Protocol) a behívásos gépeknél használható gyors módszer, a QMQP pedig a csak levélfogadásra beállított Mini-Qmail rendszerek leveleit fogadja. A sendmailen keresztüli küldéssel nincs semmi gond, automatikusan működik. A problémát az SMTP démon (qmail-smtpd) okozza. A Qmailt eredetileg úgy tervezték, hogy az SMTP portot az inetd(1) kezeli, ám a legjobb megoldás a tcpserver(1) használata, ami az ucspi-tcp csomagban található. Inetd esetén a /etc/inetd.conf file-ba írjuk be az alábbi sort:

smtp stream tcp nowait qmaild /usr/sbin/tcpd /var/qmail/bin/tcp-env /var/qmail/bin/qmail-smtpd

Tcpservert pedig így kell indítani:

tcpserver -u `id -u qmaild` -g `id -g qmaild` smtp /var/qmail/bin/qmail-smtpd &

Ha használjuk az rcpthosts file-t (márpedig miért ne használnánk), akkor – inetd használatakor – írjuk be a /etc/hosts.allow file-ba a

qmail-smtpd: [ip cimek]: setenv RELAYCLIENT sort.

Tcpserver használatakor pedig a

[ip címtartomány]:allow,RELAYCLIENT="" sort kell beszúrni a /etc/tcprules.d/smtp.conf file-ba. Ebből még létre kell hozni az adatbázisfile-t a

tcprules /etc/tcprules.d/smtp.cdb /etc/tmp < /etc/tcprules.d/smtp.conf paranccsal. Ennek megkönnyítése érdekében az általam készített rpm fileban, a /etc/tcprules.d könyvtárban, egy Makefile gondoskodik arról, hogy az átállított konfigurációs file-okból egy 'make' létrehozza az adatbázisfileokat.

Ha szeretnénk elindítani a többi (POP3, QMTP, QMQP) protokollt is, akkor hasonlóképpen kell cselekednünk ott is. A CD-mellékleten megtalálható példa rc file-ok jól szemléltetik a módszert. Miután a levelet megkapja a qmail-smtpd (vagy a többi hasonló program, mint pl.: qmail-qmqtd), a qmail-queue berakja a feldolgozási sorba. Biztonsági okokból ezt a listát már más felhasználóval, más processz futtatja. Ettől leválasztva a qmail-send (természetesen harmadik felhasználói jogosultságokkal) figyeli a beérkezett leveleket, és szortírozza. Érdemes megfigyelni a működését.

Bontsuk fel a <címzett> e-mail címét <név> és <cím> részekre.

A <cím> bent van a control/locals-ban?

Igen: → helyi továbbítás

<címzett> bent van a control/virtualdomains-ben?

Igen: → helyi továbbítás, az <e-mail> megkapja az előtétet

Ciklus (amíg a <cím> tartalmaz valamit):
A <cím> bent van a control/virtualdomains-ben?
Igen: → helyi továbbítás, az <e-mail> megkapja az előtétet
A <cím>-ből vágjuk le a gépnevet az első pontig (pl.: valami.domain.hu → domain.hu) → Távoli továbbítás
A helyi leveleket a qmail-lspawn, a távoliakat a qmail-rspawn kapja, melyek sorrendben a qmail-local, illetve a qmail-remote programokat indítják el továbbításkor. Ezeket is érdemes közelebbről megvizsgálni.

Qmail-remote:
Bontsuk fel a <címzett> e-mail címét <név> és <cím> részekre.
A <cím> bent van a control/smtproutes-ban? Igen: SMTP=>új cím (+portcím, ha meg van adva)
Nem: SMTP=><cím>, port: 25 (SMTP)
Van DNS (MX vagy A) bejegyzés az SMTP címre?
Igen: SMTP küldés
Nem: bounce (visszapattanó levél)

Qmail-lspawn:
Bontsuk fel a <címzett> email címét <név> és <cím> részekre.
A <név> bent van az users/cdb (a felhasználói lista) file-ban? Igen: → adatok a users/cdb-ből
Nem: → adatok a 'qmail-getpw <név>' programból

Qmail-getpw:
Bontsuk fel a címzett nevét <user>-<toldalék> részekre. Van <user> nevű felhasználó?
Igen: A saját könyvtára látható, és saját nevén van? Igen: → a felhasználónak → az alias felhasználónak

Qmail-local:
A HOST, HOME, USER, LOCAL, RECIPIENT, DTLINE, SENDER, RPLINE, UFLINE, EXT, EXT2, EXT3, EXT4, HOST2, HOST3, HOST4 változók feltöltése
A <toldalék>-ban az összes . kicserélése :-ra.
A megfelelő .qmail file keresése:
Van .qmail-<toldalék>? (ha nincs toldalék, akkor .qmail)
Igen: megvan
Ciklus: <toldalék> végéről levágni a szöveget az első - jelig. Van .qmail-<toldalék>-default?
Igen: megvan, a levágott toldalékot a DEFAULT változóba tölti
Nincs meg.
Ha nincs meg, hibaüzenet (No mailbox here).
A <toldalék> vége „-owner"?
Igen: Van .qmail-<toldalék>-default?
Igen: NEWSENDER = <user>-<toldalék>-owner-@<cím>-@[]
Nem: NEWSENDER = <user>-<toldalék>-owner@<cím>
A .qmail file feldolgozása (a parancsok a sor elején vannak): „#": megjegyzés, átugorja „." vagy „/": file-ba
Ha futtatható a .qmail, akkor hibaüzenet (Forward only).
Ha „/" a vége, akkor maildir, egyébként mailbox „|": programba. Ha futtatható a .qmail, akkor hibaüzenet (Forward only).
Ha a program 99-cel tér vissza, a feldolgozás megáll, ha 100-zal, akkor hiba: vissza a feladónak.
„+list": innentől kezdve csak levéltovábbítás (Forward only) „&" vagy egyéb: levéltovábbítás (forward).

A Qmail tulajdonságai

A Qmailt az írója teljesen az RFC-k alapján írta, nem törődve a nem szabványos, ám más szoftverek által engedélyezett „szolgáltatásokkal". Ezek közé tartozik például az, hogy az MTA-k többsége megengedi a magányos <LF> sorvégeket a levélben.

Ezt a Qmail nem engedi meg, de ha mégis múlhatatlan szükségünk van rá, akkor a qmail-smtpd programhívást egy sh -c 'fixcr|qmail-smtpd're kell cserélni, ami beszúrja az esetlegesen kihagyott <CR> karaktereket. A fixcr az ucspi-tcp csomagban található. A qmail másik fontos tulajdonsága, hogy nem a megszokott /var/spool/mail könyvtárban vannak a postaládák, hanem a felhasználó saját könyvtárában.

Természetesen ez nem kötelező, hiszen az eredeti procmail azokat a leveleket, amelyek nem akadtak fenn egyik szűrőn sem, a /var/spool/mail/<név> postaládába továbbítja. Ha mégis a változás mellett döntünk, akkor a levelező programokat (pl.: mailx, elm, pine, a POP3 és IMAP-kiszolgálókat) úgy kell beállítanunk, hogy a ~/Mailbox-ot használják a /var/spool/mail/<név> helyett.

A legtöbb programnak elég annyi, hogy átállítjuk a $MAIL változó értékét, de pl. a pine-nak be kell állítani az „inbox-path=Mailbox" értéket. Természetesen a legjobb választás a maildir, hiszen

számos problémát kiküszöböl (pl.: több számítógépen megosztható a felhasználó postaládája NFS-en, nem szükséges lemásolni a postaláda teljes tartalmát a POP3 kiszolgálónak stb.), de ez azzal is jár, hogy a régi levélkezelő programokat kidobhatjuk.

A CD-mellékleten megtalálható a pine4.04 és a procmail programok maildiresített változata, és a qmail tartalmazza a POP3 kiszolgálót. Van egy átmeneti megoldás is a pine, mailx és elm használatra, amik a qmail csomaggal jönnek együtt: sorban pinq, qail, elq. Ezek csak annyit tesznek, hogy létrehoznak egy ál mailboxot a maildirből a felhasználó könyvtárában, és úgy indítják a megfelelő programokat.

A procmaillel van még egy gond: a .qmail, ha programnak adjuk át a levelet, nem szúrja bele a kezdeti „From „ (a levelek közti választóvonalat), a „Delivered-To:" és a „Return-Path:" sorokat, hanem helyette környezeti változókban adja át ezeket a programnak.

Ezeket a sorokat a qmailhez adott preline nevű program írja hozzá a levélhez, így a procmail indítása a „|preline procmail" sorral történhet. A procmaillel további turpisságokat is el lehet követni.

Az EXITCODE ügyes megválasztásával elő lehet idézni a levelek automatikus visszaküldését vagy többszöri feldolgozását. Például legyen a .qmail-defaultban ez a két sor:

```
|preline procmail
&<telefonszám>@sms.pgsm.hu
```

Próbaképpen legyen a .procmailrc ez:

```
EXITCODE=100
:0
* ^Subject:.*error.*
| echo Ez hiba volt. 1>&2
EXITCODE=99

:0
* ^X-BeenThere:.*ujvilag.*
mail/ujvilag
EXITCODE=0

:0
Maildir/
```

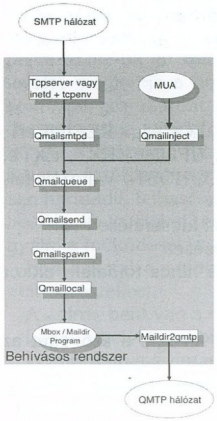
Ez azt eredményezheti, hogy az összes Errort tartalmazó témájú levél visszapattan, „Ez hiba volt" hibaüzenettel, az Újvilág levelezőlistáról érkező levelek csendben az ujvilag levelesládába mennek, míg a többi, nem szortírozott levél elmegy a megfelelő telefonszámú mobilra SMS-ként.

Nem kívánt levelek

Más néven spam. A spamszűrést már számos Internet-szolgáltatás elősegíti. Ezek az RBL-ek (Real-time Blackhole List, tehát valós idejű elnyelő lista). Ilyen a MAPS vagy az ORBS. Ezekbe a listákba vagy azokat a címeket veszik be, amelyekről kiderült, hogy spamet küldenek (MAPS), vagy azokat, amelyek ismert nyilvános levelező-átjárók (open relay). Mivel ez a lista az Interneten azonnal szabadon hozzáférhető, egyértelmű, hogy így minden beérkező levelélküldési szándékot le lehet vele ellenőrizni. Ez az ellenőrző program nincs beépítve magába a levél fogadó (smtpd) programba, több okból kifolyólag: először is így nemcsak a Qmail, hanem más, régebbi MTA is „megfejelhető" vele. Másodszor DJB szerint (D. J. Bernstein, a program írója): „az RBL a ma módszere a mai spam megállítására, a Qmail pedig egy régóta meglevő szabvány megvalósítása". Az rblsmtpd is megtalálható a CD-mellékleten, forráskódban.

Behívásos rendszerek

Nincs minden rendszer egész nap a hálózatra kötve, de ezeknél a rendszereknél is érthető követelmény, hogy érkezzenek rájuk levelek és lehessen is róluk küldeni. A problémát csak fokozza, hogy a MAPS és ORBS listákban a behívásos címek mint nyilvános átjárók szerepelnek, hiszen sosem lehet tudni, hogy éppen milyen gép csatlakozik az Internetre arról a címről. Emiatt a behívásos gépnek mindenképpen a behívó állomást kell használnia a levelek továbbítására. A qmailhez kapcsolódó serialmail csomag hivatott megoldani a levélküldés problémáját. A kimenő leveleket egy maildirbe gyűjti, és csatlakozáskor mindet elküldi vagy az SMTP, vagy a QMTP protokollon keresztül. A rendszer felépítése a 2. ábrán látható.



Így már érthető, hogy az 1. ábra QMTP bemenete az ilyen behívásos rendszerekről a központi gépre történő levéltovábbítást oldja meg. Az ilyen kapcsolatoknál azért érdemes az SMTP helyett a

QMTP-t használni, mert az az összes levelet egy kapcsolattal küldi el, míg az SMTP minden egyes levél elküldése után lebontja a TCP-kapcsolatot, a rengeteg bontás és kiépítés pedig sok időt felemészt.

A levélfogadást legkönnyebben POP3-kliensekkel oldhatjuk meg, amik közül többek között a Hacker Zsargon Szótár (The Jargon File) szerkesztése okán elhíresült Eric S. Raymond által írt fetchmail emelkedik ki. A fetchmail képes az úgynevezett multidrop postaládák kezelésére (egy domaincím összes levele egy levelesládába érkezik, és a letöltéskor válogatják) is. A példa kedvéért legyen egy nagyon.messzi.hu levelezőszerver, amit mondjuk az isp.hu neve, mint kiszolgáló alá regisztráltak. Minden levél, ami a nagyon.messzi.hura érkezik, az isp.hu kap meg. Ott beállítanak egy virtuális gépet:

/var/qmail/control/virtualdomains:
nagyon.messzi.hu:nmesszi.

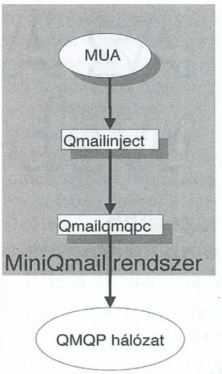
Létezik egy nmesszi felhasználó, a nagyon.messzi.hu leveleinek kezelésére, ahol az összes levelet beleirányítják a maildirbe:

/home/nmesszi/.qmail-default:
./Maildir/.
A kliens oldalon a fetchmail beállítása:
.fetchmailrc:
poll isp.hu protocol pop3 forcecr
envelope Delivered-To: qvirtual nmesszi-
user nmesszi pass <jelszó>.

Így minden csatlakozáskor a kimenő leveleket a maildir2qmtp, a bejövőket pedig a fetchmail paranccsal lehet útjukra bocsátani.

Továbbító rendszerek (mini-qmail)

A továbbító rendszerek a nagy terheltségű hálózatoknál vagy az SMTP-kiszolgálók központosításánál játszhatnak szerepet. Ezeknek a rendszereknek nincs más dolguk, mint a bejövő leveleket továbbítani egy központi szétosztó rendszerbe. A továbbító és szétosztó rendszer egymással a QMQP protokollon keresztül kommunikál. Ezen a protokollon keresztül jut be egy már feldolgozott levél egyenesen a feldolgozási sorba. A nagy terheltségű rendszerek esetén van értelme ilyen Qmail rendszer felállításának. Ott ugyanis egy néven több számítógép is létezik egyszerre (több A rekord mutat egy névre), hogy a terheltséget megosszák egymás között. Ilyenkor vagy az NFS-sel megosztott hálózatot használják (tovább lassítva az amúgy sem sokat pihenő gépet), vagy



mindegyikre Mini-Qmailt telepítenek, és a leveleket egy külön rendszer kézbesíti. Ezek a gépek akár egyenrendszerek is lehetnek, például hálózaton megosztott háttértárral, felhasználók nélkül, a biztonságot jelentő tűzfalon kívül. Mivel a rendszer szétszedhető, nagyon egyszerű elemekből áll, a bővítése nem okozhat sok problémát.

Éppen ezért még rengeteget lehetne írni róla: a saját, roppant hatékony levelezős listájáról, az ezmlmről, vagy a virtuális POP3 kiszolgálóról. Ezekkel foglalkozik a qmailnek szentelt weblap is, ahol rengeteg további ötletet, programot, patchet szerezhetünk.

Ám egyvalamit érdemes tudni a Qmaillel kapcsolatban: az alkotó, D. J. Bernstein nem szeret kétséget hagyni afelől, hogy a qmail stabil, mindig ugyanolyan jól működő szoftver. Éppen ezért csak olyan bináris Qmail-disztribúciót engedélyez, ami nem tartalmaz semmiféle változtatást, vagy legalábbis csak olyat, amit ő ellenőrzött. Ezért a CD-mellékleten található Qmail rendszer nem tartalmaz semmilyen kiegészítést. Az általam karbantartott krondor.kva.hu (www.kva.hu) rendszeren már több mint fél éve hibátlanul működik ez a program. Egyszerre száz levél küldését bonyolítja le (ez valószínűleg még kevés is a forgalomhoz képest), és megerőltetés nélkül szolgálja ki az általános szolgáltatásokon (pl.: www, ftp, DNS, TACACS, Ipchains) kívül az Rpg, SRun, Gimp, Ujvilag, Iksz, Rlug, Buli és Szissz levelezőlistákat. Maga a gép egy 2.2.1-es Kernelt futtató Pentium 200-as, 32 MB memóriával. ■

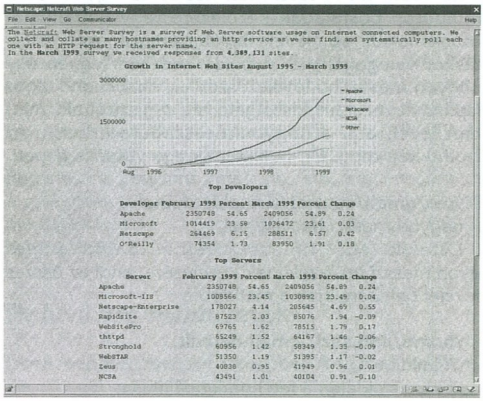
MUTATÓ

Források: http://www.qmail.org
Magyar tükör: http://www.agria.hu/qmail
D. J. Bernstein oldalai: http://pobox.com/~djb
EZMLM (a Qmail levelezőlistája):
http://www.ezmlm.org
Qmail levelezőlista: qmail@list.cr.yp.to
Feliratkozás: qmail-subscribe@list.cr.yp.to
Leiratkozás: qmail-unsubscribe@list.cr.yp.to

Czakó Krisztián
Slapic@vogel.hu

Az Apache webszerver

A Linux egyik legnépszerűbb felhasználási területe az internetes szerverek üzemeltetése, azon belül is elsősorban a webszervereké. Erre a célra több szoftver is rendelkezésünkre áll. A legnépszerűbb egyértelműen az Apache, mely – Linuxon és más operációs rendszereken futó változatait számbavéve – a világ webszervereinek több mint 60 százalékán üzemel.



Megpróbálunk ízelítőt adni az Apache képességeiből, illetve alapvető telepítési és beüzemelési tanácsokkal szeretnénk olvasóinkat ellátni, Debian Linux-alapokon. Amiért a profi linuxosok jelentős része kedveli a Debiant, az a csomagkezelés és a csomagok kiváló integráltsága. Egy, az Apacheéhoz hasonlóan komplex rendszer telepítésekor nem mindegy, mennyi pluszmunkát jelent egy-egy új funkció vagy kiegészítő telepítése. Hála a Debian fejlesztőinek, ez nem okoz gondot.

Mit tegyünk fel?

A 2.1-es Debian óta az Apache alapjai is több különálló csomagban kapnak helyet. Észrevehetjük, hogy amikor a dselect segítségével kijelöljük az Apache csomagot, a függőségkezelés azonnal szól, hogy szükség van az apache-commonra is. Külön csomagba kerültek ugyanis az Apache alapjai és a binárisok. Ez akkor jó, ha az alap binárisokhoz képest mást, például az SSL-t támogató változatot szeretnénk feltenni. Hála az Egyesült Államok törvényeinek, az SSL-t használó programok nem lehetnek részei az Egyesült Államokban koordinált Debian disztribúció alaprészének. Ezért készül hozzá egy úgynevezett non-US rész, melyet Európában koordinálnak, és amit így nem érintenek az USA megszorításai. Természetesen a CD-mellékleten ez is megtalálható. Az Apache alapkiépítésben „mindössze" egy igen sokoldalú webszerver. Ennél azonban sokkal többre is képes. Ha például felteszünk mellé egy adatbázisszervert és a PHP3 modulokat, máris lehetőségünk lesz szerveroldali adatbáziskezelő-programok futtatására.

Beállítások

Mint a legtöbb Debian csomaghoz, az Apache-hez is van egy debianos beállítóprogram, mely sima szöveges felületen tesz fel kérdéseket, amire válaszolva egy számunkra ideálisan beállított webszerverhez jutunk. Minden beállítást a /etc/apache vagy – az SSL-es változat esetén – a /etc/apache-ssl könyvtárban találunk. Az ott található lényeges file-ok:

– a httpd.conf, melyben a webszerver alapvető beállításait és a virtuális szerver beállításokat találjuk;

– az access.conf, melyben a hozzáférési jogosultságok beállításai;

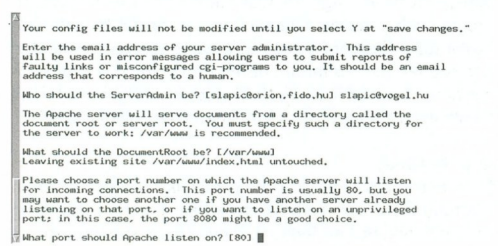
– az srm.conf, ahol a MIME és néhány különlegesség beállítása található.

Szükséges, hogy legyen ebben a könyvtárban egy mime.types file, amiben a MIME-típusokat kell felsorolni. Szerencsére már létezik egy ilyen a Debianban (a /etc-

ben), ami egyszerűen csak ide van linkelve. Nem kifejezetten az Apache-hez tartozik a cron.conf, ez egy debianos extra. Az Apache működése során vannak rendszeresen elvégzendő feladatok. Ezeket indítja Linux alatt a cron daemon. Debian alatt, az Apache-hez tartozó karbantartó szkript innen veszi a beállításait, azaz itt határozhatunk meg néhány alapvető dolgot. Ezek a következők: hány napra visszamenőleg tárolja a naplófile-okat (APACHE_OLD_LOGS), mely napokon fusson a szkript (APACHE_DAYS_TO_RUN), valamint milyen programot indítson el a naplófile-ok forgatása (azaz a régiek mentése) után (APACHE_POST_SCRIPT). Ez utóbbit használhatjuk például a naplófile-ok analizálására. A felsorolt többi konfigurációs file-ról majd később írunk. Az említett beállító szkriptet apacheconfig-nak, SSL-es Apache esetén apache-sslconfig-nak hívják. Ezek automatikusan elindulnak, amikor először telepítjük az Apache-szervert, de utána rootként bármikor újra lefuttathatjuk őket, ha valamit módosítani akarunk. Intelligens, tehát nem bántja az általunk kézzel elvégzett módosításokat, csak azokhoz nyúl, melyekre rá is kérdez.

Ha frissítjük az Apache-t, vagy modult teszünk fel hozzá – ami külön csomagban van – a modul vagy az Apache csomag telepítés utáni szkriptje rákérdez, hogy megfelelnek-e az aktuális beállítások, vagy újra akarjuk konfigurálni a programot. Ez utóbbira leginkább egy új modul telepítésekor van szükség, hogy bekerüljenek a modulhoz szükséges sorok a konfigurációba. Futása végeztével mindig rákérdez, hogy újraindítsa-e a webszervert.

Most lássuk az apacheconfig kérdéseit! Az első kérdés, hogy ki a szerver adminisztrátora. Itt egy létező e-mail címet kell megadnunk. Ezt például olyan hibaüzenetek aljára fogja írni, melyek beállítási hiba miatt keletkeztek. A dokumentumok gyökérkönyvtára a következő kérdés. Ez lesz az a könyvtár, ahol alapesetben az Apache keresni fogja a dokumentumokat, azaz innen indul a webszerver könyvtárstruktúrája. Természetesen virtuális-szerverek használata esetén ezt – csakúgy mint a legtöbb beállítást –, virtuális-szerverenként külön meg tudjuk adni (de ez nem kötelező). Debian alatt a /var/www az alapértelmezett beállítás, amit nem érdemes módosítani, szerintünk jó választás. Ha a webszerver adatait nem akarjuk a /var partícióján tárolni, könnyedén csatolhatunk egy másik partíciót vagy file-rendszert /var/www néven, amint azt a telepítésről szóló írásunkban már említettük (igaz nem pont erre e könyvtárra, de analóg módon alkalmazhatók erre is az ott leírtak). Itt, ha még nem volt index.html file a megadott könyvtárban, létrehoz egyet a program, ha volt, csak annyit közöl, hogy nem bántotta. A következő lépés azon port megadása, amin a webszerverünknek várnia kell a kapcsolatokat. Normálisan ez a nyolcvanas port, minden böngésző itt keresi a szervert. Lehetnek speciális esetek, ahol más portra akarjuk tenni, de tehetünk virtuális-szervereket is más-más portra. Ezzel a trükkel olyan gépen is futtathatunk Apache-webszervert, ahol nincs root jogosultságunk, azaz nincs lehetőségünk privilegizált portra (1024 alatti portok) „tenni" egy szervert. Ebben az esetben például a 8080-as portra téve működni fog az Apache. Az ilyen címekre a böngészőben a portszám megadásával kell hivatkozni. Például ha a www.noname.hu szerveren üzemeltetünk a 8080-as porton egy Apache-t, az ezt elérő URL így fog kinézni: http://www.noname.hu:8080/.



Ide tartozik még, hogy ha SSL-t használunk, az SSL-es http port a 443-as, de ezt nem itt, hanem majd kézzel az apache-ssl beállításai között kell megnézni (természetesen az apache-ssl nem csak SSL-es kiszolgálást tud, ezért nem kell mellé sima Apache csomag is).

Talán az egyik legfontosabb és leghasznosabb lehetőség beállítása, a modulok konfigurálása következik. Pontosabban azt dönthetjük el, hogy mely modulokat használja a telepítettek közül az Apache és melyeket ne. Volt már szó a modulokról, de arról még nem, mik is ezek. A kernelhez hasonlóan, az Apache is képes megfelelően elkészített külső file-t betölteni, mely a saját kódjába épül a memóriában. Az itteni modulok annyiban eltérnek a kernelben használatosaktól, hogy csak induláskor tölthetők be, futás közben nem. Tehát az itt megadott modulok betöltődnek a szerver indulásakor, és végig foglalják a memóriát. Tehát szemben a kernel moduljaival – amik gondosan ügyelnek a memória megőrzésére azáltal, hogy csak szükség esetén töltődnek be –, az Apache esetében csak a rendszeres újrafordítását spóroljuk meg a modulok használatával.

Ha csak egy új funkció kell, feltesszük a modult tartalmazó Debian csomagot, vagy – ha nincs Debian csomag – fordítunk egy modult, újrakonfiguráljuk az apacheconfiggal a szervert (vagy kézzel beírjuk a megfelelő file-okba, ha az jobban tetszik), újraindítjuk az Apache-t és kész. Ugyanezzel az egyszerűséggel távolíthatunk el egy-egy modult: kiveszszük a betöltését szolgáló sort (LoadModule) a konfigból, és az újraindított Apache máris kevesebb memóriát használ. Egy-egy ilyen újraindítás mindössze 2–3 másodperc, így csak apróbb kényelmetlenséget jelentenek azoknak, akik éppen weblapunkat nézegetik. Tehát a modulok kiválasztása: első lépésben eldönthetjük, hogy egyenként, kézzel akarjuk-e kiválasztani, hogy a telepített modulok közül melyike-

ket használja a szerver, vagy meghagyjuk az alapértelmezett beállításokat. Ha nincs szükség speciális funkciókra, általában az alapértelmezett megoldás jó. Ellenkező esetben kézzel kell kiválogatni a modulokat. Természetesen ezeket a beállításokat megtaláljuk a httpd.conf file-ban (LoadModule).

Itt most nem ismertethetünk minden modult részletesen, de egy részükről a későbbiekben szó fog esni. Annyit fontos megjegyezni, hogy a „required" jelöléssel ellátott modulokat mindenképpen kapcsoljuk be, mert szükségesek a szerver normális működéséhez (csak nagyon speciális esetben hagyhatók el). A „standard" jelölésű modulok azok, melyek az általános használat közben kelleni szoktak, de nem alapvető fontosságúak, míg a többiek azok az extra funkciók, melyre vagy szükségünk van – vagy nem. Ha ezzel elkészültünk, a beállítószkript kis számolás után összeállítja az új beállításokat, melyeket ha kérünk el is ment. Ha nem kérjük, minden amit itt beállítottunk elvész. A változások mentése után a következő kérdés arra irányul, hogy az új beállításokkal újra szeretnénk-e indítani az Apache-t. Ezzel el is készültünk egy egyszerű vagy – ha sok modult tettünk fel és kapcsoltunk be – igen komplex webszerver telepítésével.

Minden további beállítást az említett file-okban tudunk elvégezni. Talán érdemes annyit ehhez hozzáfűzni, hogy nem muszáj a végletekig duzzasztani a konfigurációs file-okat, fel is darabolhatjuk őket, és hivatkozhatunk a külön néven elmentett részekre a megfelelő standard config file-ból. Például sok virtuális szerver esetén lehet szükség a szétbontásra – érdemes különvenni az egyes virtuális szerverek beállításait. E módszerrel könnyebbé tehetjük a megfelelő beállítás vagy paraméter megtalálását és módosítását. Ennél van egy még sokkal trükkösebb lehetőségünk is, mégpedig az, ha a beállításokat egy adatbázisban tartjuk. Innen egy – az Apache Perl modulja segítségével a szerveren futó – Perl program olvassa ki és hajtja végre őket. Ezzel a módszerrel akár egy teljesen webes alapú konfigurálófelület is készíthető hozzá.

Az alábbiakban megpróbáljuk összefoglalni a főbb beállítási opciókat. Semmiképpen nem vállalkozhatunk a száznál is több lehetőség részletes bemutatására – erre egy könyv is kevés lenne –, de segítséget szeretnénk nyújtani az alapvető működéshez szükséges funkciók beállításához. A teljes (angol nyelvű) dokumentációt megtaláljuk a www.apache.org-on, illetve az apache-docs Debian csomag telepítése után a /usr/doc/apache/ könyvtárban HTML-formátumban. A három említett alap-konfigurációs file közül kezdjük a httpd.conf-fal! Mint írtuk, ebben vannak az alapvető opciók.

ServerType

Itt azt határozhatjuk meg, hogy milyen módon használjuk az Apache-t. Két választásunk van: „standalone" vagy „inetd". Az utóbbi beállítás esetén, hogy az inetd „superdaemon" fogadja a kapcsolatokat, és minden egyes kapcsolatra elindít egy Apache-t. Ezt csak akkor javasoljuk, ha a webkiszolgálás nem gépünk elsődleges funkciója, csak néha-néha van rá szükség. Ekkor feleslegesen futna és foglalna memóriát az Apache. Ha azonban egy rendszeresen látogatott webszervert építünk, az inetd-s megoldás kifejezetten erőforrás pazarlóvá válik, mivel minden egyes új kéréshez egy új Apache-t kell indítania. Ez a megoldás ráadásul lassabb is, mint a másik, ahol az Apache folyton fut, és ő maga indít „gyermek" (child) processzeket. A gyerekek indulása, mivel az a fő-Apache már fut, csak minimális extra erőforrást igényel. Ráadásul egy-egy ilyen gyermek elindulása után nem csak egyetlen, hanem számos (e szám beállítható) kérést ki tud szolgálni egymás után.

E megoldás hátránya az, hogy folyamatosan igényel memóriát. Ez a ritkán használt szerver esetén nem szerencsés (bár a Linux szépen kiteszi a swapbe az épp nem futó Apache processzeket, teljes memóriafoglalásukkal egyetemben). A legegyszerűbb (és mi is ezt ajánljuk), ha maradunk a standalone módnál. Ekkor az Apache Debian alatt a „/etc/init.d/apache start" paranccsal indítható, és a „/etc/init.d/apache stop" paranccsal állítható le, beállításait pedig a „/etc/init.d/apache reload" parancs hatására fogja újraolvasni. Természetesen a megfelelő /etc/rc?.d (a kérdőjel itt egy számot jelent nullától–hatig, ami a Linux megfelelő futási szintjét jelenti) könyvtárba helyezett szimbolikus link elvégzi az automatikus indítást és leállítást az operációs rendszer indításakor és leállításakor. Megjegyezzük, hogy ezek a Debian csomagból feltett Apache esetén jól be vannak állítva, nincs vele dolgunk.

Port

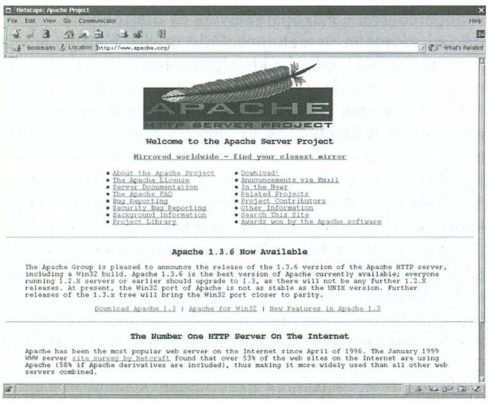
Erről már volt szó az apacheconfig leírásánál. Itt adjuk meg a portszámot, amin a szerver „figyel".

HostnameLookups

Az IP-kapcsolatok esetén a hívót mint IP-számot kapja meg az a szerver, amelyik fogadja a kapcsolatot. Ez bőven elég a szerver számára, de nem túl sokatmondó az emberi szemnek. Ezzel szemben egy név már igen, és mint tudjuk, az Interneten az IP-számokhoz nevek (és viszont) tartoznak. Ha szeretnénk, hogy a naplófile-ba tárolható legyen a hívó gép bejegyzett neve, vagy azt CGI szkriptekben és beágyazott programokban felhasználhassuk, ezt az opciót be kell kapcsolni (on). Hatására a szerver, a name-szerver megkérdezi a kapott IP-számhoz tartozó nevet. Amiért ezt ki is lehet kapcsolni az, hogy az ilyen visszakérdezések időigényesek, ami lassíthatja a kiszolgálást és feleslegesen terhelheti a szervert.

User és Group

Ez a két opció határozza meg, melyik felhasználó jogán fog futni a szerver. Természetesen az Apache-t lehetőség szerint rootként kell indítani, illetve ha a standard 80-as



porton (vagy bármely más 1024 alatti porton) szeretnénk használni, mindenképpen rootként kell indítani, hiszen csak a root user jogosult az úgynevezett privilegizált, azaz 1024 alatti portok használatára. Az Apache-nek ezen, és az esetleg root tulajdonban lévő naplófile-ok írásán kívül nincs már szüksége a root jogaira, így azt ha itt beállítjuk, „leadja". Ez pontosan annyit jelent, hogy a rootként elvégzendő feladatok után a tényleges user-id-t az itt megadottra cseréli. A Group opció ugyanez, csak a csoportra. Debian alatt mind a user, mind a group a „www-data" user, mely szerepel is a felhasználók, illetve csoportok listájában. Érdemes ezt a beállítást meghagyni. A root jog meghagyását biztonsági okokból semmiképpen sem javasoljuk.

ServerAdmin

Szintén az apacheconfig-nál írtunk erről az opcióról: a szerver üzemeltetőjének e-mail címe.

ServerRoot

A szerver úgynevezett root, azaz gyökérkönyvtára. Nem keverendő a dokumentumok gyökérkönyvtárával. Ha az Apache-t forráskódból telepítjük, minden komponense – ideértve a beállításokat, naplófile-ok, webdokumentumok helyét – egyetlen könyvtárba kerül, és elvileg ez a „ServerRoot". Debianban ez picit más, mondhatnánk logikusabb, illeszkedik a Linux file-rendszer felosztásába. Mivel a szerver minden beállítását az itt megadott könyvtárban keresi (kivéve természetesen azt, amelyikben e könyvtárat megadjuk – hiszen azt vagy a fordításkor, vagy parancssorban tehetjük meg), pontosabban az ebből a könyvtárból nyíló conf alkönyvtárban (ez Debian alatt csak egy szimbolikus link a .-ra, ami az aktuális könyvtár, azaz a /etc/apache/conf ekvivalens a /etc/apache-val), itt a /etc/apache-t kell megadni, ha nem módosítottuk a beállítások helyét. Természetesen ez az alapértelmezés látható a telepített konfigban is.

BindAddress

Itt adhatjuk meg, hogy a gépünk hálózati címei közül melyeken „figyeljen" az Apache-szerver. Ez annyit jelent, hogy az itt megadott címeken érhető el szerverünk. Ennek egyik fő előnye az IP alapú virtuális-szerver üzemeltetés. A régebbi HTTP protokoll nem ismerte a név szerinti virtuális szerver fogalmát, így ezzel csak az IP szerinti virtuális-szerver-üzemeltetés működött. Mégpedig úgy, hogy a szerver gépnek több IP-száma volt (ez lehetett egy vagy több hálózati csatolón), és minden IP-számra elindult egy-egy önálló webszerver, mely a megfelelő IP-szám szerint figyelt. Ha az ő általa kiszolgált IP-számra jött a kapcsolat, ő felelt. Ma már szerencsére ismeri a HTTP protokoll a név szerinti virtuális szervert, és azzal sokkal egyszerűbb az ilyet megvalósítani (erről még lesz szó). Az alapértelmezett „*" azt jelenti, hogy minden létező címünkön elérhető a szerverünk.

LoadModule és AddModule

Erről is volt már szó korábban az apacheconfig leírásánál. Egy LoadModule sor tölti be a paraméterében meghatározott modult. Ebből többet is felsorolhatunk, minden sorban egy-egy modul betöltését előírva a szervernek. Az első paraméter a modul neve, a második annak az osztott könyvtárnak (shared library) a neve, melyet be kell tölteni. Az AddModule speciális modulok betöltéséhez használatos, erre példa az SSL-t megvalósító modul. Formátuma egyszerűbb: csak a modul nevét kell megadni.

ErrorLog és TransferLog

A két alapvető naplófile helyét és nevét adhatjuk meg. Az ErrorLog a hibaüzeneteket, míg a TransferLog a kiszolgált anyagok és az adatot lekérő adatait tartalmazza. Ez utóbbi statisztikák készítésére használható, de lehetőség van egyedi naplófile generálására, ahol szabadon definiálhatjuk azt is, mi és hogyan kerüljön bele. Debianban a naplófile-ok (log-ok) a /var/log könyvtárba kerülnek, ahol ha szükséges, egy-egy szervernek saját alkönyvtára van. Az Apache azok közé tartozik, akiknek van, méghozzá Apache néven. Ebbe kerülnek a naplófile-ok, ha nem módosítottuk a beállításokat. A hibák az error.log-ba, a forgalom az access.log-ba kerül.

PidFile

Linux alatt illik, hogy minden szervernek legyen egy ilyen, úgynevezett „pid" file-ja. A pid, azaz process-id a futó szerver azonosítója, illetve ha több processze fut, akkor az elsődleges processzé. Ezt az id-t használhatjuk, ha jelzést (signal) szeretnénk küldeni a szervernek. Ilyen jel-

zéseket használunk a leállítására, vagy arra, hogy újraolvastassuk vele a beállításait. A pid file-ok szokásos helye a /var/run könyvtár, ezen belül a szerver nevét használjuk .pid kiegészítéssel. Azaz Apache pid file-ja a /var/run/apache.pid. Ezt csak akkor változtassuk meg, ha tudjuk mit csinálunk!

ServerName
Itt adhatjuk meg szerverünk nevét. Nem feltétlenül kell megadni. Ha nem adunk meg értéket, az Apache indulásakor megkérdezi a name-szervert, hogy mi is a saját neve. Persze ez sokszor rossz, mivel gyakran nem azt a nevet, hanem egy www kezdetű nevet szeretnénk a webkiszolgálónknak adni. Ilyenkor ezt írjuk be az opció paramétereként. Figyeljünk, hogy a DNS-be bejegyzett valódi név kerüljön ide, mivel ellenkező esetben problémáink lehetnek, jobban mondva a böngészőnek lesznek problémái, ha a szervertől egy ilyen névvel ellátott hivatkozást kap, és azt nem tudja elérni. Ez az opció is használható a virtuális szerverek egyedi beállításainál, így itt csak az alapértelmezettet adhatjuk meg.

CacheNegotiatedDocs
Manapság egyre inkább dinamikusan épülnek fel a weboldalak, azaz egy-egy oldal tartalma futás közben keletkezik programok kimeneteként. Az ilyen dokumentumok tárolása a proxy-cache szerveren általában nem kívánatos, és ezért az Apache küld egy „Pragma: no-cache" üzenetet a dokumentummal. Ha mégis szeretnénk, hogy az ilyen lapokat tárolja a proxy-cache, írjuk be ezt a sort, vagy szedjük ki a kommentet (#) előle.

Timeout
Az időtúllépés határértékét állíthatjuk be ezzel. A paraméterben magadott másodperc letelte után minősít sikertelennek egy adott átvitelt (adat fogadást vagy küldést) a szerver. Az alapértelmezés 1200, azaz 20 perc. Ennél vehetjük lejjebb is.

KeepAlive
A Keep-Alive a kérések maximális száma kapcsolatonként. Normálisan, egy-egy dokumentum átadása után a HTTP kapcsolat megszakad, azaz minden egyes dokumentum letöltéséhez új kapcsolat épül fel. Ezt változtatja meg a Keep-Alive-kérés, melyet a kliens küld a szervernek, hogy az ne bontsa a kapcsolatot. Ezen kérések maximálisan elfogadott számát állíthatjuk be. Ha nullára vesszük, kikapcsoljuk ezt a funkciót (Nem javasolt!).

KeepAliveTimeout
A fenti opció párja, csak engedélyezett Keep-Alive esetén van értelme. Megadhatjuk, hány másodpercet várjon a szerver Keep-Alive kapcsolat esetén a következő kérésre.

MinSpareServers, MaxSpareServers és StartServers
Három szorosan összekapcsolódó opció. Mint írtuk, az Apache minden kapcsolat kiszolgálásához egy-egy új gyermek processzt indít, és a kapcsolatot átadja azoknak. Továbbá, hogy ezek a gyermekek nem állnak le azonnal, hanem tovább is kiszolgálják az újonnan beérkező kapcsolatokat. Azaz, ha van szabad gyermek, a szerver annak ad-a át a kérést, és csak akkor indít újat, ha nem talál szabadon várakozót. A kiszolgálás sebességének növelése érdekében az Apache indulásakor „unatkozó" gyermekeket indít, hogy ha jön a kapcsolati kérés, azonnal ki tudja szolgálni, ne kelljen várni a gyermek elindulására. Természetesen szükség esetén futás közben is indít ilyen tényleges kérést nem kiszolgáló processzt, ha már nincs ilyen. A MinSpareServers határozza meg, hogy legalább hány ilyen unatkozó gyermeknek kell futnia, a MaxSpareServers pedig azt, hogy maximum hány futhat. A tényleges szám ezen két érték között az igények szerint változik.

A StartServers az indulási értéket határozza meg: ennyi gyermeket fog indítani az Apache amikor elindul.

MaxClients
Itt maximalizálhatjuk az egyszerre kiszolgálható kliensek számát. Azaz ennél több párhuzamos kérést nem fog kiszolgálni a szerverünk. Értékét úgy határozzuk meg, hogy a rendszert ne terhelhesse túl sok egyszerre érkező kérés. Túl alacsonyra se vegyük, hiszen akkor sokan maradnak hoppon.

MaxRequestsPerChild
Egy-egy gyermek processz által maximálisan kiszolgálható kérések száma. Ha ezt a gyermek elérte, leáll, és ha kell, helyette a szülő egy új processzt indít. A periodikus leállítással biztosíthatjuk, hogy ha hibásan működik valamelyik gyermek (vagy az általa indított program), ne fogyaszthassa el az összes memóriát. Linuxon nagyon stabil az Apache, nem fog problémát okozni, ha ez a szám magas.

ProxyRequests
Egy Apache-szerver, ha a proxy modult betöltöttük, alkalmas proxy-cache-ként is üzemelni, azaz a böngésző és a tényleges webkiszolgáló közötti adatokat továbbítani. Erre csak kisebb helyeken lehet szükségünk, ahol az Apache által nyújtott egyszerűbb proxy-szolgáltatások elegendőek. Komolyabb helyekre inkább a Squid proxy-cache szoftver használatát javasoljuk. A funkciót az „on" opcióval lehet be- és az „off" opcióval kikapcsolni. Ha a cache funkcióra is szükségünk van a proxy mellett, használjuk a „Cache" kezdetű opciókat, melyeket itt nem ismertetünk.

Listen
A már beállított cím és portszám mellett további címeken és portokon is „figyelhet" a szerverünk. Ezeket adhatjuk itt

meg. Ez is jól használható virtuális-szerverek építésére (más port – más virtuális szerver), de az SSL-es változatnál például így adhatjuk meg, hogy használja a https 443-as portját.

NameVirtualHost
A manapság használt leggyakoribb virtuális-szerver megvalósítás. A HTTP 1.1-es protokoll a szerverhez menő kérésben nemcsak azt adja át, hogy melyek dokumentumot szeretné megkapni, hanem azt is, mi a hívott szerver neve. Erre azért van szükség, mert a szerver automatikusan csak a hívott IP-számot kapja meg a kapcsolatban, ami esetünkben mindig ugyanaz. A kommunikáció során átadott szervernév lehetővé teszi, hogy a szerveren más-más dokumentumot adjunk ki ennek alapján. Ahhoz, hogy ez működjön, először is kell a NameVirtualHost bejegyzés, melynek paramétere azon IP-száma a szervernek, melyen működnie kell a virtuális-szervereknek. Ezután következhetnek az egyedi virtuális-szerverek bejegyzései.

A bejegyzések az alábbi formátumot követik:<VirtualHost IP-szám:port> virtuális-szerver beállításai </VirtualHost> Az első bejegyzés „IP-szám"-mezője az az IP-szám, melyet a NameVirtualHostban megadtunk, vagy ha ott többet adtunk meg, akkor az, melyen az adott virtuális-szervernek ki kell szolgálnia a kéréseket. A „port" az a portszám, melyen figyelni kell. Alapértelmezés szerint, ha a„:port" részt elhagyjuk, ugyanaz, amin a szerver figyel (Port opció). Ha például SSL-es szervert építünk, az SSL-t kiszolgálni egy ilyen virtuális-szerverrel lehet, melyben csak annyival térünk el az alapbeállításoktól, vagy a párjaként használt, nem SSL-es virtuális-szervertől, hogy beírjuk az „SSLEnable" opciót (csak SSL-es Apache-vel működik!).

Ezzel már ejtettünk is pár szót a virtuális-szerver beállításairól. Itt lényegileg ugyanazok az opciók alkalmazhatók, mint a „normál" szervernél, azaz megadhatjuk a ServerAdmint, a ServerName-et, a ServerAliast (azon nevek listája, melyre ez a virtuális-szerver „hallgat"), a DocumentRootot, mely talán a legfontosabb: így vehetjük máshonnan az adott virtuális-szerver által kiadott dokumentumokat, így lehet más kezdőlapot adni az egy szerveren működő két névnek. Amit még érdemes idetenni, az az ErrorLog és TransferLog opciók, hogy külön naplózzuk a virtuális-szerverek forgalmát, ezzel is megkönnyítve a későbbi statisztikák elkészítését. Ezzel a httpd.conf alapvető beállításaival megvagyunk. Mint azt jeleztük, a szerver ennél nagyságrendekkel több beállításra képes.

Folytassuk a sort az access.conf alapjaival. Ide kerülnek a jogosultsági beállítások. Megadható, hogy mit és honnan lehessen elérni, milyen feltételekkel.

Kétféle korlátozás lehetséges: a könyvtárra, valamint a helyre (location). Fontos megemlíteni, hogy ezen beállítások is, mint majdnem minden opció kerülhet egy-egy virtuális-szerver beállításai közé.

Könyvtárak

Ahhoz, hogy bármi elérhető legyen a szerverünkön, legalább egy könyvtár elérését engedélyezni kell, nevezetesen a DocumentRootban beállítottét. Egy könyvtár meghatározása az alábbi módon történik: „<Directory könyvtár>", ahol a „könyvtár" az szabályozni kívánt könyvtár. A /var/www-re állított DocumentRootnál maradva először egy „<Directory /var/www>" sornak kell következnie. Ezután jöhetnek az opciók, majd a végén a „</Directory>" sor. Az opciók az alábbiak lehetnek:

Options
Ez a legáltalánosabb, alapvető opciók meghatározása. Itt szabályozhatjuk, hogy lehessen-e könyvtár-listát kérni, ha nincs index.html file a könyvtárban (Indexes), hogy követse-e a szimbolikus linkeket (FollowSymlink), lehessen CGI-t futtatni (ExecCGI), használhatunk-e beszúrt file-okat (Includes), vagy épp ezek összessége (All), vagy egyike sem (None).

AllowOverride
Az srm.conf beállításai, mint például a fenti opciók bizonyos feltételekkel felülbírálhatók a .htaccess file-ban. A feltételeket itt adhatjuk meg, a .htaccess file-t pedig abban a könyvtárban kell elhelyezni, ahol felül akarunk bírálni valamit. Ugyanazokat az opciókat használhatjuk, mint az srm.conf-ban. Egy tipikus felülbírálás az, amikor azonosító és jelszó megadásához kötjük egy adott weblap letöltését.

A felülbírálás engedélyezéséhez használható opciók: All, azaz minden felülbírálható, Options, azaz a fenti opciók bírálhatók felül, FileInfo, azaz a file-információk, AuthConfig, azaz a hozzáférés-beállítások vagy a Limit, azaz a korlátozások bírálhatók felül. Óvatosan bánjunk ezek engedélyezésével, mert könnyen adhatunk olyan jogokat, amiket eszünk ágában sem volt.

order, allow, deny
Ezekkel határozzuk meg, ki kaphat adatot a szervertől. Az order paramétereként jogosultságokat adhatunk meg, mint az „allow", azaz engedélyezett és a „deny", azaz tiltott. Az „allow,deny" sorrend esetén először az „allow" opciókat értékeli ki a szerver, és csak utána a „deny"-t, miközben az alapértelmezett jog a tiltás. A „deny,allow" esetén előbb értékeli a deny-t, utána az allow-t és az alapértelmezés az engedélyezés.

Van egy harmadik opció, ez a „mutual-failure". Ez akkor, és csak akkor engedélyezi az elérést, ha a host szerepel az „allow" listán, és nem szerepel a „deny" listán. Mindig az adott esetre kell kiválasztani a megfelelőt. Az „allow" opció adja meg, hogy mit engedélyezünk, a deny pedig, hogy mit tiltunk. Például egy „allow from all" opció lehetővé teszi, hogy min-

denhonnan elérhető legyen az adott hely, ha a /var/www-s példánál maradunk, akkor a webszerverünk kezdőlapja. Az „allow from" paramétereként megadhatunk IP-számot, hostnevet, részleges hostnevet vagy IP-szám/netmask formátumban subnetet is a fenti all mellett. Ugyan ezeket használhatjuk a „deny from" esetében is. Íme egy példa, hogyan tegyük lehetővé a /var/www/intranet elérését mindenkinek a .vogel.hu domain alá, míg mindenki másnak tiltjuk:

```
<Directory /var/www/intranet>
order deny,allow
deny from all
allow from .vogel.hu
</Directory>
```

Egy példa az Apache egy speciális lehetőségére, amivel lekérhetjük a weben keresztül szerverünk állapotát. Ehhez a „serverstatus"-kezelőt fogjuk használni, mely modulban áll rendelkezésünkre (feltételezzük, hogy a modult bekapcsoltuk). Mivel ehhez nem egy könyvtár, hanem egy URL használatát kell engedélyeznünk, a „<Location hely>" opcióval kell megadni, ahol „hely" az URL-beli relatív elérési rész (a hostnév után). Tehát a példa:

```
<Location /server-status>
SetHandler ser-ver-status
order deny,allow
deny from all
allow from admin.vogel.hu
</Location>
```

A fenti jelentése, hogy a /server-status hivatkozást (függetlenül attól, hogy előtte milyen név szerepelt) a server-status funkció kezeli (SetHandler), a jogosultságok kiértékelését a tiltásokkal kezdjük, az alapértelmezés engedélyezett (order deny,allow), megtiltjuk mindenkinek az elérést (deny from all), majd engedélyezzük az elérést az admin.vogel.hu nevű gépnek (allow from admin.vogel.hu).

Bizonyos hibákra és esetekre megadhatunk egyedi hibaüzenetet a beépített egyszerű üzenetek helyett. Egy érdekes példa erre, hogy a hibakezelést továbbítjuk egy másik szervernek. Valamikor az 1.1-es verziónál korábbi Apache-szerverekkel együtt terjesztettek egy phf szöveggel kezdődő nevű szkriptet, melyben volt egy kellemetlen biztonsági hiba. Az alábbi példa, mely a minta access.conf file-ban is olvasható, átirányítja az ennek szóló kéréseket (potenciális betörési kísérleteket) a http://phf.apache.org/phf_abuse_log.cgi URL-re, mely naplózza az eseményt.

```
<Location /cgi-bin/phf*>
deny from all
ErrorDocument 403
http://phf.apache.org/phf_abuse_log.cgi
</Location>
```

Ez előzőekhez képest itt az ErrorDocument opció az újdonság. Ennek használata egyszerű: „ErrorDocument hibakód szöveg-vagy-URL". A hivatalos hibakód, melyet az adott hiba esetén a szerver ad, a szöveg-vagy-URL pedig az a szöveg, amit a szerver ki kell írjon, vagy az az URL, amit be kell töltenie. Itt használhatunk egy szkriptet is, ami legenerálja a HTML lapot. Azaz így tudunk dinamikus, kért URL és konkrét eseményfüggő hibaüzeneteket generálni, mely több, mint a megszokott egysoros tömör üzenet. Nem az access.conf része, de szó volt a .htaccess alkalmazásáról, essék hát pár szó használatáról. Mint írtuk, oda kell elhelyezni, ahol hatását érvényesíteni szeretnénk, és a fent leírt, illetve más jogosultságokat szabályzó opciókat használhatunk benne.

A legegyszerűbb, ha ezt is egy példán keresztül mutatjuk be: AuthUserFile /etc/apache/htpasswd AuthName „Belépés" AuthType Basic <LIMIT GET POST> require valid-user </LIMIT>Itt az eddigiekhez képest csak újdonságok vannak. Mint látható, nem szerepel sem a Directory, sem a Location jelzés, mivel az opciók arra a könyvtárra vonatkoznak, melyben a .htaccess file található. Az első sor (AuthUserFile) egy hivatkozás arra a file-ra, melyben a felhasználók neve és jelszava szerepel. A példa szerint ez a /etc/apache/htpasswd, de bárhová tehetjük ezt a file-t. Egyvalami nem javasolt: betenni a letölthető dokumentumok közé, tehát oda, ahol a .htaccess is van, hiszen akkor (ugyancsak bizonyos feltételekkel) letölthető lesz. A jelszó-file tartalma formailag egyszerű, és azt a htpasswd parancscsal készíthetjük el. A „htpasswd -c jelszófile név" létrehozza a „jelszófile" jelszó-file-t, felveszi bele a „név" azonosítót, amihez megkérdezi a jelszót, és beírja azt kódolt formában. Ha a file már létezik, a „-c" opciót hagyjuk el. A „htpasswd /etc/apache/htpasswd slapic" parancs hozzáadja a slapic felhasználót a /etc/apache/htpasswd file-hoz, és bekéri az új jelszavát, majd azt is beteszi az említett file-ba. Természetesen, ha a felhasználó már létezik, a parancs a jelszavát megváltoztatja.

Az „AuthName" a következő opció. Ennek mindössze annyi a feladata, hogy megadja, mit írjon ki a böngésző, hogy mihez kéri a jelszót. A „Limit" opció meghatározza, hogy mely funkciók, jelen esetben a küldés és letöltés (GET, POST) legyenek korlátozva, majd megadjuk, hogy mi az elérés feltétele (limit és /limit közti rész), ami most csak annyi, hogy létező felhasználó, azaz olyan valaki, aki szerepel a /etc/apache/htpasswd-ben, és sikeresen azonosította magát. A „/limit" jelzi az adott korlátozás leíró szakasz végét.

Az srm.conf

Eljutottunk az utolsó konfigurációs file-hoz, az srm.conf-hoz. Ide kerültek a szolgáltatásokat érintő beállítások.

DocumentRoot

Az apacheconfig ismertetésénél már volt róla szó, ez határozza meg, mi legyen a szerver által kiszolgált gyökérkönyvtár. A javasolt és alapértelmezett könyvtár Debian alatt a /var/www.

UserDir
Ha nem tiltjuk meg, a rendszerünkön minden felhasználó készíthet saját honlapot. Bizonyára sokan találkoztak már a www.valahol.hu/~felhasznalo típusú címekkel, amiből nekünk most a „~felhasznalo" rész a fontos.

A „~" a Unix rendszerekben a „home" könyvtár jele. Ez az a könyvtár, ahol a felhasználó a saját adatait tárolja. Ezért jelölik a weben is ugyanígy a felhasználó saját könyvtárát, ami természetesen nem lesz azonos a tényleges „home"-mal.

Pontosabban, ha itt megadunk egyetlen könyvtárnevet (például „public_html"), akkor a felhasználó „home" könyvtárából nyíló „public_html" könyvtárat fogja jelenteni. Picit konkrétabban ez azt jelenti, hogy ha a Slapic felhasználó „home" könyvtára a /home/slapic, és a www.vogel.hu szerveren vagyunk, akkor a www.vogel.hu/~slapic a /home/slapic/public_html könyvtárból adja az információkat, ha a könyvtár létezik. Ellenkező esetben hibaüzenetet kapunk, hogy ez a cím nem létezik. A példában feltételeztük, hogy a UserDir opciót paramétereként a public_html szöveget írtuk be, ami a legáltalánosabb hely Unix rendszereken.

DirectoryIndex
Ha az URL-ben csak könyvtárnevet adunk meg, és nincs file-név, a szerver minden esetben az itt megadott file-t vagy file-okat keresi a megadott sorrendben, és amint megtalál egyet, azt fogja átadni a böngészőnek. Alaphelyzetben ez az „index.html", de beírhatjuk például az „index.php3"-at is, minek hatására ha nincs index.html, de van index.php3, akkor ez utóbbi jelenik meg.

FancyIndexing
Ha engedélyeztük a könyvtárlisták lekérését valahol, és az adott könyvtárban nincs olyan file, mely szerepelne az előbb ismertetett DirectoryIndex opcióban, a szerver ad egy listát a könyvtárban lévő file-okról. Ha a FancyIndexing opciót bekapcsoljuk (on), a lista egy picit jobban lesz díszítve, mint egyébként. Például típustól függő ikonok bukkannak fel a file-nevek mellett.

AddIconByEncoding, AddIconByType, AddIcon
A fent említett ikonok hozzáadásának kontrollálására használjuk ezeket opciókat. Valószínűleg a „gyári" srm.conf-ban lévők jók, és nem sok módosítani valónk akad rajta, így ezt bővebben nem is részletezzük.

AccessFileName
A hozzáférési jogoknál volt szó a .htaccess file-ról. Ott nem említettük, hogy a „.htaccess" név csak az alapértelmezés, ezzel az opcióval megváltoztathatjuk. Nem érdemes viszont ezt megtenni, ha nincs rá különösebb okunk.

AddLanguage, LanguagePriority
A szerver és a böngésző párbeszédükben azt is megbeszélik, hogy elsősorban milyen nyelvű dokumentumot szeretne a felhasználó látni.

Az Apache lehetővé teszi, hogy ugyanarra a névre vagy könyvtárra hivatkozva más-más dokumentumokat adjunk át ettől a beállítástól függően. Ezek az opciók szabályozzák, hogy mely nyelveket támogatjuk (AddLanguage), és ha nem ad meg a böngésző nyelvet – vagy az adott anyagból nem létezik a megadott nyelvű, mely nyelveket keresse a szerver.

Alias
Ezzel adhatunk meg olyan hivatkozásokat, melyek teljesen máshova mutatnak, mint a hivatkozott könyvtár vagy file. A „gyári" srm.conf-ban található minta pont jó ennek bemutatására:Alias /icons/ /usr/share/apache/icons/Jelentése, hogy a „/icons/" hivatkozás esetén a /usr/share/apache/icons/ könyvtárt nézi, nem a /var/www/icons/-t (feltételezve, hogy a DocumentRoot /var/www).

ScriptAlias
Az előző ponthoz hasonló, csak a CGI (Common Gateway Interface) szkriptekre vonatkozik. Azt határozza meg, hogy hol találjuk a szerveren a CGI szkripteket.

AddType
Újabb MIME-típusok hozzáadására alkalmazhatjuk. Mint azt írtuk, Debian Linuxban a MIME-típusok leírása az /etc/mime.types file-ban található, mely az Apache részére symlinkelve van az Apache konfigurációs könyvtárába. Ha ebben a file-ban nem szerepel a szükséges MIME-típus, vagy oda írjuk be, vagy – ha csak az Apache-nél van rá szükség – ide, az AddType opcióval. Javasoljuk, hogy mindenki ez utóbbi megoldást válaszsza, mert a /etc/mime.types-t jobb automatikusan frissíteni amikor az azt tartalmazó Debian csomagot frissítjük.

Az opció formátuma nagyon egyszerű: „AddType típus végződés".

A típus a MIME-megnevezés, azaz típus/altípus (pl. text/html) a végződés pedig a file vége (pl.: .html).

AddHandler
A fentihez hasonlóan MIME-típusokat és végződéseket kapcsol össze, de ennek célja, hogy kezelőt (handler) definiáljon egy-egy konkrét végződéshez, hogy azt a megfelelő modul kezelhesse.

Ezzel az Apache főbb opcióinak a végére értünk. Remélhetőleg segítettünk megérteni a szerver működését és alapvető beállításait. Itt nem állt módunkban ennél bővebb leírással szolgálni. Az érdeklődőknek bátran merjük ajánlani az igen bő és részletes elektronikus dokumentációt. ■

Magosányi Árpád
mag@bunuel
.tii.matav.hu

Az új generációs syslogd

A Unix rendszerek hibaüzeneteit és az eseményekről szóló jelzéseket a „syslog" nevű alrendszer gyűjti össze. Ennek legfontosabb programja a syslog démon, ami a neki átadott naplóbejegyzéseket állományokba írja, és/vagy továbbítja más rendszerekhez, felhasználókhoz.

A syslog démon minden Unix rendszernek része, és az alapvető igényeknek igen tisztes kora ellenére még ma is megfelel. Vannak azonban olyan alkalmazásai is, ahol az eltelt évek megmutatkoznak.

Az egyik ilyen terület a hálózaton keresztül történő eseménygyűjtés; a klasszikus syslogd protokollja nem teszi lehetővé, hogy a bejegyzéseket megbízható és hiteles módon juttassuk el egy másik gépre, és megakadályozzuk azt, hogy illetéktelenek azokat meghamisítsák.

A másik probléma az, hogy a klasszikus syslogd nincs extrém nagy forgalmak kezelésére kihegyezve, hiszen egy nagyobb gép naplóforgalma is maximum néhány megabyte, és csak egészen speciális hálózatvédelmi alkalmazásoknál kell több száz megabyte-ot naplózni naponként.

További nagy probléma, hogy az üzeneteket nem tudja a mai kor kihívásainak megfelelő flexibilitással szűrni. Néhány előre definiált csoportba sorolja őket, és néhány előre megadott prioritást rendel hozzájuk. Szűrni csak ezek szerint lehet.

Ezeknek a problémáknak a megoldására született meg az új generációs syslog, a syslog-ng. Ez a program mindent tud, amit a klasszikus syslogd, és azon felül néhány extrát is, méghozzá olyan módon, hogy nagyon nagy forgalmak kezelését és továbbítását is képes legyen elvégezni kevés erőforrás felhasználásával.

Fontos tudni, hogy ez a program GPL alá eső szabad szoftver, és egyike azoknak a szabad szoftvereknek, amelyeket üzleti alapokon készítettek; azaz a a fejlesztő pénzt kapott érte.

A syslog-ng konfigurálása különbözik a klasszikus syslogd konfigurációjától, hiszen sokkal több dologra képes. Viszont a konfiguráció nem annyival bonyolultabb, mint amennyivel többet tud a program. A szoftver használatához fontos ismerni annak alapelveit. A syslog-ng különböző forrásokból vár naplóbejegyzéseket, azokat opcionális módon különféle szűrőkön engedi át, majd különböző célokhoz továbbítja őket. A források és célok az alábbiak lehetnek:

- file: az üzenet a megadott file-ba íródik, vagy onnan kerül olvasásra. Ha a file-név nem „/"-rel ezdődik, akkor a logdir által megadott könyvtárat használja. Ha célt definiálunk, a file-nak léteznie kell.
- tcp: digitálisan aláírt TCP/IP-n átvitt üzenet. Ehhez szükséges egy megosztott titok mind a forrás mind a cél oldalon, aminek az options-ban megadott könyvtárban kell lennie. (alapértelmezés /etc/syslog-ng)
- udp: a klasszikus syslogd-vel kompatibilis UDP feletti protokoll. Általában az 514-es porton szokott menni.
- unix-dgram: Datagram unix socket. Sok Unixban helyi hostról ilyen módon jönnek a logok, paraméter egy file-név.
- unix-stream: Stream unix socket. Linuxon a helyi hostról így jönnek az üzenetek, paraméter egy filenév, Linuxon /dev/log.
- user: a megadott felhasználó termináljára kiíródik az üzenet.
- |parancs: a parancs szabványos bemenetére kerül az üzenet.
- sun-stream: Streams socket. Régebbi Solarisokon ilyen socketet használ a klasszikus syslogd.
- sun-door: A Sun doors socketje. Újabb Solarisokon használatos.

Egy forrás megadása például a következő módon történik: `source classical { unix-stream /dev/log; udp 0.0.0.0,514; };`

Ez a „classical" nevű forrást definiálja, ami egy unix stream socketet jelent a /dev/log útvonallal, és minden hálózati interfészen UDP hálózati protokollt az 514-es porton. Linuxon a klasszikus syslogd éppen ezeken a forrásokon figyel.

Egy célt például a következő módon adhatunk meg: `destination remote { tcp loghost.domain.org,514 ;};`

```
# options
options { sync 100; mark 1200; directory /etc/sys-
log-ng; logdir /var/log; };
# csak a példa kedvéért minden opció szerepel.

# sources
source local { unix-stream /dev/log; };
source remote { tcp 0.0.0.0,514; };
# itt egy lokális és egy távoli logforrást definiáltunk,
# a távoli logforrás az új protokollt használja.

# destinations
destination messages { file /var/log/messages; };
destination console { file /dev/tty11; };
destination authlog { file /var/log/auth; };
destination loghost { tcp loghost.domain.org,514; };
# ltt két állományba, egy virtuális konzolra
# és egy távoli gépre kerülnek a logok

# filters
filter f_info { priority info; };
filter f_notice { priority notice; };
filter f_warn { priority warn; };
filter f_auth { facility != auth; facility != authpriv; };
filter f_match { match-prog „sendmail"; match „RSA
key generation complete"; };
# itt néhány prioritás szerinti,
# egy kicsit bonyolultabb csoport szerinti
# egy program és minta szerinti szűrést láthatunk.

# log statements
log { source local; filter f_info, f_notice, f_warn; filter
f_auth; destination messages; };
log { source local; destination console; };
log { source local; filter f_auth; destination authlog; };
log { source remote; filter f_match; destination log-
host; };
# itt látható a definiált források, szűrők és célok
# egymáshoz rendelése.
# fontos szempontot mutat be az első példa,
# amelyben a három vesszővel
# elválasztott prioritásokat figyelő szűrő
# vagylagos kapcsolatát
# kötjük össze a csoportra vonatkozó szűrővel
# egy „és" kapcsolattal.
```

Ez a „remote" nevű célt definiálja, ami a loghost.domain.org nevű host 514-es portján figyelő, a syslog-ng fejlettebb protokollját használó syslog démonnak küldi tovább az üzeneteket.

Mint említettük, különféle szűrőket is meg lehet adni, hogy a feltételektől függően az üzeneteket más-más helyekre irányíthassuk. Ezek a feltételek az alábbiak:

- facility: megadott csoportba (facility) tartozó üzenet
- priority: megadott prioritású üzenet
- facpri: a fenti kettő együtt
- match: megadott reguláris kifejezésre illeszkedő bejegyzések
- match-file: megadott állományban található minták valamelyikét tartalmazó üzenetek
- match-prog: megadott programtól származó üzenetek.

Egy szűrőt például a következő módon lehet megadni:
filter f_auth { facility != auth; facility != authpriv; };

Ez az „f_auth" nevű szűrőt definiálja, amin azok az üzenetek mennek át, ahol az üzenetcsoport sem az „auth" sem az „authpriv" csoporttal nem egyenlő.

A fentieken kívül különféle opciókat lehet definiálni, amelyek a program működésének egyéb aspektusait befolyásolják. Jelenleg a következő opciók léteznek:

- sync: a megadott számú sor kiírása után az output file-okat synceli, ez az egyes file-oknál külön-külön is beállítható.
- mark: a megadott masodperc idő letelte után egy – MARK – sort küld a célokra
- directory: ebben a könyvtárban keresi a kulcs-file-okat
- logdir: ebbe a könyvtárba kerülnek a logfile-ok, ha a filenév nem „/" -rel kezdődik.

Egy opciókat definiáló sor például a következő:
options { sync 100; mark 1200; directory /etc/syslog-ng; logdir /var/log; };

Mindezeket a forrásokat, szűrőket és célokat valamiképpen össze kell tudni kötni. Ezt a célt szolgálja a „log" kifejezés, amely a források, szűrők, célok és esetleges opciók felsorolásából áll.

Erre egy példa a következő:
log { source src; filter f_auth; destination f_auth; };

Ez a kifejezés azt mondja, hogy az „src" forrásból származó üzenetek közül azokat, amik átesnek az f_auth szűrőn, az f_auth célba kell eljuttatni.

A mellékelt dobozban egy teljes konfigurációs állományt is olvashatunk.

Aki többet szeretne megtudni a programról, a http://www.balabit.hu/ weboldalon talál információkat, illetve működik egy levelezőlista is. Erre a http://www.vekoll.vein.hu/mailman/listinfo/syslog-ng oldalon lehet feliratkozni. ■

Balázs-Csiki László
bcsl@elender.hu

Pehelysúlyú programozás Unixon

Számomra a legcsodálatosabb dolog a Linuxban az, hogy szinte bármit akarok elérni, van megoldás, sőt több megoldás is létezik. Ráadásul nem az, hogy lebutított és/vagy harminc napig működő programokat próbálok letölteni, hanem a számos szabadon terjesztett programnyelv segítségével magam írom meg a kis programjaimat.

Többnyire csak összeillesztem a dolgok nehéz részét elvégző komponenseket. Azok az interpretált nyelvek, amelyekről a továbbiakban szó lesz olyanok, hogy gyorsan és élvezetesen mehessen rajtuk a programfejlesztés: a változókat nem kell deklarálni, a típuskonverziók automatikusak, a programot nem kell minden változtatás után lefordítani, a reguláris kifejezések vagy az asszociatív tömbök sok algoritmust egyszerűvé tesznek. Hátrányuk viszont, hogy lassabbak mint a lefordított (például C vagy C++ nyelvű) társaik, és nagyobb programok esetén a típusok nagyvonalú kezelése (például a stringek szükség esetén automatikusan konvertálódnak számokká) megbosszulhatja magát. Sokszor mindegy, hogy egy program egy századmásodperc alatt fut le, vagy esetleg egy fél másodpercig is elvacakol, az viszont nagyon is számít, hogy fél óráig vagy öt percig tart-e egy adott probléma megoldása.

Szükség esetén viszonylag könnyen meg lehet oldani, hogy a program időigényes része C/C++-ban legyen írva, a többi meg egy szkriptnyelven.

Legtöbbet a profik által gyártott szkriptekből tanulhatunk. Sok Unix parancs szkript: a /bin , /usr/bin , /sbin/ vagy a /usr/sbin könyvtárakban a file * | grep script paranccsal válogathatjuk ki a szkripteket. Az /etc/rc.d/ könyvtár rendszerinicializációs szkriptjei is nagyon tanulságosak.

Futtatás

Egy shellszkriptet többféleképp lehet futtatni.

1. Ha egy pontot (vagy a source parancsot) teszel a file-név elé, akkor az aktuális shell soronként olvassa be a file-t, és a hatás ugyanaz, mintha te gépelted volna be ezeket a sorokat is.
2. Ha futtatási jogot adsz a file-ra (chmod +x file-nev), akkor egy új shell indul a szkript futtatására.
3. Akárcsak akkor, ha expliciten egy új shellnek adod át paraméterként.

A nem-shell szkriptek esetén csak a második és harmadik módszer áll rendelkezésre, míg a harmadik esetben értelemszerűen a megfelelő fordítónak kell átadni programkánkat. A második módszer a legjobb általában: a szkript parancsként viselkedik, és mivel külön processz lesz, mindenféle dolgokat végezhetsz vele: a háttérben indíthatod (ha utána teszel egy & jelet), felfüggesztheted (Ctrl-Z), újraindíthatod a háttérben (bg), illetve az előtérben (fg). Ilyenkor a file első sorában a #! karakterek után a fordító teljes elérési útvonalát kell megadni.

A #!path_a_programhoz jelölés

Ha egy file nevét minden további nélkül beírod a parancssorba, akkor a következő dolgok történnek.

1. A shell ellenőrzi, hogy a file létezik-e, és hogy van-e futtatási jogod rá.
2. Ha igen, akkor továbbadja a file-t a kernelnek, aki az első néhány byte-ot beolvasva eldönti, hogy hogyan futtassa. Ha ezek a byte-ok ELF vagy a.out típusú bináris futtatható állományra utalnak, akkor az ennek megfelelő lépések történnek. Ha egy Java class file, és a Java biná-

jelölések/szkriptnyelv	sh	csh	tcl	perl	python
program neve	$0	$0	$argv0	$0	sys.argv[0]
első argumentum	$1	$1	lindex $argv 0	$ARGV[0]	sys.argv[1]
összes argumentum	$*	$*	$argv	@ARGV	sys.argv
argumentumok száma	$#	$#argv	$argc	@ARGV (skalarkontextusban)	len(sys.argv) - 1
listán végigmenő for	for	foreach	foreach	foreach	for
lépésenként menő for	-	-	for	for	-
feltétel	if-then-fi (elif)	if-then-endif (else if)	if{}{}	if() {} (elsif)	if: (elif)
interaktív input kérése	read INPUT	set VALASZ=$<	gets stdin VALASZ	$input=<STDIN>	raw_input()

risok közvetlen futtatásának támogatása bele van fordítva a kernelbe, akkor a file átadódik a Java fordítónak. Végül (és ez az eset, ami érdekes számunkra), ha a file a #! karakterekkel kezdődik, akkor a kernel a mi file-unkat a #! után következő file-nak adja át argumentumként. Tehát ha van egy huhu nevű file-unk, ami a „#!/usr/bin/perl -w" sorral kezdődik, akkor a kernel az „/usr/bin/perl"-t futtatja „-w huhu" argumentummal. Minden valamirevaló unixos szkriptnyelvben a # után következő rész megjegyzésnek számít, vagyis ez az első sor nem interferál a szkript tartalmával. Kedvencem az öngyilkos „rm szkript", ami a #!/bin/rm sorral kezdődik. Az olvasóra bízom a további érdekes (és néha még hasznos) lehetőségek felderítését. Mivel a kernel az, aki a #! karakterek utáni parancsot keresi, nem elég, ha a beírt file-név benne van a shell által (tehát egy szinttel magasabban) használt PATH-ban. Például a #!perl nem fog működni. Az egyes nyelvek fordítóinak helye változhat Unixról-Unixra, ami problémát okozhat: ha a szkript, amit az Internetről letöltesz #!/usr/local/bin/perl -el kezdődik, és nálad csak /usr/bin/perl van, akkor „command not found" hibaüzeneteket kapsz. Mindenképpen érdemes a következő linket létrehozni:

ln -s /usr/bin/perl /usr/local/bin/perl

Egy másik megoldás, ami univerzálisabb, de valamivel lassabb: #!/bin/sh-val kezdik a programot (ami mindig létezik), aztán a shellre bízzák a fordító megtalálását és elindítását. A futtatható szkriptjeid legjobb helye a home könyvtárad bin/ könyvtárában van. Ellenőrizd, hogy a bin/ könyvtárad benne van-e a keresési útvonalban (echo $PATH)!

Shellváltozók

Mint hamarosan látni fogjuk, a shell nemcsak átadja a parancsokat a kernelnek, hanem egyben programozási nyelv is, változókkal, vezérlési szerkezetekkel. A környezeti változók olyan speciális shellváltozók, amelyeket a shell által futtatott programok látnak. Ezek többnyire a shellinicializációs file-okban (például bash esetén /etc/profile ~/.bash_profile ~/.bashrc) kapnak értéket.

Példák:

HOME – ahová az argumentum nélküli cd visz

PATH – azoknak a könyvtáraknak a neve kettősponttal elválasztva, amelyekben a shell a futtatható parancsokat keresi

PS1 – a prompt string

Gyakran csak arra használják a shellszkripteket, hogy elindítsanak egy másik programot (ami lehet bináris vagy egy másfajta szkript), miután beállították a környezeti változókat. Ilyenkor az execkel kell indítani a programot, hogy átvegye a processzek között a shell helyét és ne fusson mindvégig feleslegesen egy shell. Egy shell által futtatott program semmilyen körülmények között nem változtathatja meg szülő-shelljének változóit. Találós kérdés: miért nincs /bin/cd program? (a cd parancs miért van mindig a shellekbe beépítve?)

Parancssor – sed és awk

A Unix parancssor éppen olyan, mint a DOS parancssor, csak éppen sokkal többet tud. Több parancs áll rendelkezésünkre, kényelmesebben használható (például parancs- és file-név kiegészítés), valamint jól is programozható. A Unix parancsokat eleve úgy készítették, hogy az átirányítás és a pipe segítségével jól tudjanak együttműködni. Az inputjukat éppúgy tudják a standard inputról, mint file-ból szedni, az eredményüket a standard outputra írják, az esetleges hibaüzeneteket pedig a standard errorra.

Kedvencem az, amikor egyik gépről a másikra pipeolunk – a következő paranccsal küldhetjük a public_html könyvtárunkat összecsomagolva egy másik gépre:

tar czvf - public_html | ssh szervergep 'cat > public_html.tar.gz'.

A tar a standard outputra ír, ha a file-névnek egy mínuszjelet adunk. Az ssh rendes Unix parancsként akár szűrőként is használható, csakúgy mint a cat.

File-ok keresése és a Unix filozófiája

Amikor még csak ismerkedtem a Unix parancssorral, sokat bosszankodtam, hogy miért nincs a grepnek rekurzív opciója. Gyakran volt szükségem arra, hogy rekurzívan keressek egy egész könyvtárstruktúrában. grep */* -szerű dolgokkal próbálkoztam, de persze nem volt jó, mert nem tudtam, hogy hány alkönyvtár mélységben lesz a keresett dolog. Aztán rájöttem, hogy direkt nincs. A Unix filozófiája ugyanis az, hogy ahelyett, hogy minden parancsot teleaggatnának mindenféle elképzelhető opcióval és ugyanazokat az opciókat minden parancshoz hozzáadnák, inkább minden parancs csak egy dolgot csinál (és azt jól). A parancsok együttműködésének koordinálását bízzuk a shellre és a kernelre, valamint a felhasználóra. (Akit megfelelően intelligensnek aposztrofálnak az önálló döntések meghozatalára.)

Az olyan parancsok esetén, amelyeket általában egy egész könyvtárstruktúrára akarunk ráuszítani (cp, chmod, chown stb.), van értelme rekurzív opciót adni, viszont biztosan nem akarunk könyvtárnevekben, bináris file-okban stb. greppelni. Ez a megközelítés amellett, hogy egyszerű (vagyis valószínűleg hibamentes) programokat eredményez, rendkívüli rugalmasságot nyújt a kreatív felhasználóknak, akik úgy állítják össze a parancssoraikat, ahogy nekik jólesik. Tegyük fel, hogy .pl kiterjesztésű file-ok között keresel olyat, ami a fork stringet tartalmazza. File-ok keresésére a find parancsot használjuk:

```
find . -name '*.pl' -print
```

Az aktuális könyvtárból kiindulva végignézi az alkönyvtárakat, és megtalálja az összes pl. végződésű file-t. A GNU findnak nem kell a -print opciót megadni, mert (igen logikusan) ez a default, de hordozhatónak szánt szkriptekbe be kell tenni, mert egyéb findok lehet, hogy semmit sem írnak ki nélküle. A grep fork `find . -name '*.pl' -print` megoldás a parancs behelyettesítést használja, vagyis a „visszafele idézőjelek" közé tett parancs eredménye adódik át a grepnek file-argumentumként. Ez azonban nem mindig jó, mert van egy korlát arra, hogy hány argumentum adható át egy parancsnak, és ha a find ennél több file-t talál, akkor a grep nem fogja az összeset megkapni. Ilyenkor használják az xargs nevű speciális programot, ami a standard inputjára érkező adatokat átadja az argumentumaként kapott parancsnak argumentumként: find . -name '*.pl' print | xargs grep fork. A fenti esetben, ha a find eredménye ./proba/huhu.pl és ./lassuk.pl, akkor a xargs a grep fork ./proba/huhu.pl ./lassuk.pl programot futtatja.

Az xargs kiküszöböli a korábban említett problémát: ha túl sok az argumentum, akkor többször is elindítja a programját. Aki pedig unja begépelni, az természetesen írhat egy néhány soros shellszkriptet, ami a findot pontosan úgy paraméterezi, ahogyan neki szüksége van rá. Például a biztonság kedvéért érdemes a „-type f" opciót megadni, hogy csak file-neveket adjon át a grepnek. Egyébként a Unix-filozófiával eretnek módon szakító emberek írtak rekurzív grepeket is, és vannak a windowsos 'Find File'-hoz hasonló grafikus felületű keresők is, például a KDE-ben a kfind. Ez is a find, xargs és grep programokat futtatja. Előnye, hogy nem kell opciókat megjegyezni, a hátránya viszont az, hogy nem illeszkedik a Unix-filozófiába: az eredményt nem lehet közvetlenül másik programba irányítani. A find az egyik leghasznosabb Unix parancs. A GNU find dokumentációja, sajnos elég homályos, ezért álljon itt néhány példa:
find . -atime +7 -print megtalálja azokat a file-okat, amelyek legalább egy hete nem lettek használatba véve. Ha +7 helyett -7 van, akkor azokat találja meg, amelyeket elértek az elmúlt héten.

A find / \(-type d -a -perm -007 -a -user root \) csak az olyan könyvtárakat írja ki, amelyek a root tulajdonában vannak, de mindenki létrehozhat bennük file-okat. A feltételeket csoportosíthatjuk (a „-a" az ésnek, a „-o" a vagynak felel meg), s ezt zárójellel védjük meg attól, hogy a shell (subshellként) értelmezze.
A find / \(-name core -o -name '*.bak' \) -atime +7 -exec rm {} \;
pedig letörli azokat a file-okat, amelyeknek a neve core, vagy .bakban végződik és már egy hete nem voltak bolygatva. A {} helyére sorra beíródnak a talált file-ok. Ha csak egy file tartózkodási helyére vagy kíváncsi, egy a gyökérkönyvtárból indított find helyett praktikusabb a gyors locate parancsot használni. Ez a következőképpen működik. Éjszakánként cron-ból lefut az updatedb nevű program (egy tanulságos shellscript, ami a findot futtatja megfelelően felparaméterezve, a find eredményét pedig a sort, uniq, awk, tr stb. segítségével dolgozza fel), ami egy adatbázist épít fel, és ebben már gyorsan lehet keresni.

Sed+awk

A sed, mint neve is mutatja egy Stream Editor, azaz olyan szövegszerkesztő, ami nem interaktív. Soronként olvas egy file-ból (vagy a standard inputról), ezeken a sorokon elvégzi a megadott típusú változtatásokat, majd az eredményt a standard outputra írja. Konverziós programok írására ideális. Például, ha van egy hosszú szöveged, amiben a bekezdések üres sorokkal vannak elválasztva, és HTML-formátumba akarod konvertálni, akkor a cat szoveg.txt | sed 's/^$/<p>/g' > szoveg.html automatikusan egy <p>-t tesz minden üres sorba.

Az awk 1978-ban született meg, és három megalkotójáról (Al Aho, Peter Weinberger és Brian Kernighan) nevezték el. Ez már egy igazi programnyelv, vezérlési szerkezetekkel, tömbökkel, beépített függvényekkel stb. A soronként beolvasott bemenetet rekordokra osztja, ezeket a $1, $2, $3 stb. változókba teszi, amikkel a későbbiekben azt tehetünk, ami jólesik. Például:
ls -l | awk '{print $5,$9}' az „ls -l"
kimenetéből kiírja az ötödik és kilencedik rekordot, vagyis a file-ok méretét és nevét. (Ugyanezt a du -a is megtenné, méghozzá rekurzívan, de az nem alkalmas az awk bemutatására...) Az awk alapértelmezés szerint a szóközöket és a tabulátorokat veszi rekordelválasztónak, ezt a -F opcióval lehet megváltoztatni. Például az /etc/passwd file-ban a rekordokat kettőspont választja el, így egy awk-os megoldás a jelszó nélküli felhasználók kiíratásának klasszikus problémájára: awk -F: 'length($2) == 0' /etc/passwd. Linuxon az awk egy link a gawk-ra (GNU awk), ami a „hagyományos" awk-nak egy kiterjesztett változata. A Perl előtti időkben egész bonyolult awk szkripteket volt szokás írni, de ma már jobb Perlben elkészíteni azt, ami bonyolultabb a fentieknél. Meglévő awk szkripteket az a2p programmal lehet Perlbe konvertálni.

A shellek fajtái

Kezdetben volt a Bourne-shell és a C-shell. A Bourne-shellt (sh) hatékonyabban lehetett programozni, a C-shellt (csh) viszont kényelmesebben lehetett interaktívan használni (például csak ebben volt history-mechanizmus, ami előhozza a korábban beírt parancsokat). Ennek megfelelően többnyire az sh-t használták shellszkriptek írására, a csh-t pedig interaktív gépelésre. Ez azonban elég skizofrén állapot volt, hiszen így az interaktívan felépített shellprogramokat más szintaxissal kellett írni, mint a file-ba írtakat. Ezért aztán megszületett több olyan shell is, ami az sh-val programozási szempontból felülről kompatibilis volt, de interaktívan is kényelmesen lehetett használni: ksh (Korn-shell, Linuxon pdksh néven lehet beszerezni), bash (Bourne Again-shell, a FSF „hivatalos" shellje) és a zsh (Z-shell).

Interaktív célra én a zsh-t ajánlom, egy jól konfigurált zsh szinte előre kitalálja a gondolataidat, ráadásul nemzeti büszkeségünk is dagadozhat használata közben: a fejlesztés jelenlegi koordinátora, Hídvégi Zoltán ugyanis hazánkfia. Közben a csh-nak is megjelent egy javított változata tcsh néven, ennek ellenére a csh-szintaxisban nem érdemes elmélyedni, mert „a parancssor elszáll, a szkript megmarad" törvény eredményeképp körül vagyunk véve csodálatos sh-szkriptek tömegével, és inkább ezeket érdemes megértenünk.

Ami a file-ba írt shellszkripteket illeti, a hordozhatóság érdekében azokat legjobb a #!/bin/sh-val kezdeni, és a továbbiakban is csak a Bourne-shell lehetőségeit használni. Linuxon az sh többnyire csak egy link a bash-ra, a csh pedig a tcsh-ra. A bash egy kicsit másképpen viselkedik, ha sh-ként hívod, de így is elfogad olyasmiket, amik az igazi sh-ban nincsenek meg. Linuxon ash-val lehet tesztelni, ha probléma a kompatibilitás (ash a legprimitívebb Bourne-kompatibilis shell). Nem kell feltétlenül szkripteket írni, ha ki akarjuk használni a shell programozhatóságát.

Tegyük fel például, hogy van egy rakás PostScript file-unk egy könyvtárban, és egymás után szeretnénk megnézni őket. Fáraszto és unalmas lenne minden alkalommal File/Openre kattintgatni, valamint utána kiválasztani a következőt, vagy minden file-ra újraindítani a gv-t. Ilyenkor a következőt lehet a zsh parancssorba írni: $ for i in *.ps; gv $i; vagy $ for i (*.ps) gv $i esetleg a szabványos sh alapú megoldásként $ for i in *.ps; do; gv $i; done. Feltételes elágazást a következő módon készíthetünk:

```
if feltétel
then
parancsok
fi
```

Az if az őt követő parancsok közül az utolsó kilépési értékét (exit status) nézi. Ha az nulla – vagyis a parancs sikeresen futott –, akkor igaznak veszi, különben hamisnak. (Megjegyzés: a többi logikai típus nélküli programozási nyelvben ez pont fordítva van: a nulla felel meg a hamisnak, minden más pedig az igaznak). Például:

```
#!/bin/sh
if who | grep haver > /dev/null
then
echo A haver be van jelentkezve
fi
```

A grep eredményét átirányítottuk a /dev/null-ba (vagyis átadtuk az enyészetnek), mivel csak a grep kilépési értéke érdekel bennünket: szerepel-e a „haver" string a who parancs által kiírt sorokban vagy sem. Különféle feltételek meglétét tesztelhetjük a „test" parancs segítségével:

```
#!/bin/sh
if test $# -eq 0 # argumentumok nélkül hívtak
then
echo Használat: argumentumot is kérek.
exit 1
fi
```

A „test" helyett mindig lehet két szögletes zárójelet tenni, így olvashatóbb a feltétel: if [$# -eq 0] # argumentumok nélkül hívtak.

If-szerű viselkedést if nélkül is elérhetünk, ha két parancs közé az && jelet tesszük, akkor a második csak

akkor hajtódik végre, ha az első a 0 viszszatérési értékkel jelzi sikeres futását. Például:
latex szoveg.tex && xdvi szoveg.dvi
vagy
gcc -o proba proba.c && proba
A || jel esetében pont fordítva, akkor hajtódik végre a második, ha az első nem sikerül, például:
szukszavu_parancs || echo „Valami nem stimmel" 1>&2 .

For ciklus

A for ciklus némiképp különbözik a Pascal és C nyelvek for ciklusától, mégpedig abban, hogy csak egy rögzített lista elemein lehet végigmenni vele. Tipikus felhasználása, hogy bizonyos kiterjesztésű file-okon vagy a parancssor argumentumain megyünk végig. Például a következő szkript megpróbálja törölni a home könyvtáramban található HTML file-okat, de a ~/lynx_bookmarks.html file-ra nem kérdez rá.

```
#!/bin/sh
for i in ~/*.html
do
if [ $i != ~/lynx_bookmarks.html ]
then
rm -i $i
fi
done
```

While és until

A while ciklus addig fut, amíg a „hasában" levő feltétel igaz, az until pedig pont fordítva: amíg a feltétele igaz nem lesz. Mindkét esetben do és done közé kell zárni a ciklus parancsait. A következő szkript kissé trükkösen ellenőrzi a jelszó nélküli felhasználókat:

```
#!/bin/sh
IFS=":"
while read nev jelszo uid gid gcos home shell
do
if [ -z „$jelszo" ]
then
echo „$nev -nek nincs jelszava [ $gcos ]"
fi
done < /etc/passwd.
```

A kezdete a végén van: az /etc/passwd file-t irányítjuk a while-do-done ciklusba, a read soronként olvas, és az IFS (Internal Field Separator) által elválasztott mezőket teszi a megfelelő változókba (az IFS szerepe ugyanaz, mint az awk -F opciójáé.) Olvashatóbb lett volna, ha az /etc/passwd-ot az elején írjuk be a ciklusba:

```
cat /etc/passwd | while read nev jelszo uid gid gcos home shell.
```

Ez azonban a cat külső program felesleges meghívását jelenti, ami könnyen egy századmásodperc késést is jelenthet az előző verzió futásidejéhez képest...

Házi feladat:

Gyakorlásképpen írjon minden olvasó egy kis shellszkriptet, ami ellenőrzi, hogy az X alól futtatják vagy egy normális terminálon. Tipp: X esetén a DISPLAY környezeti változó be van állítva (a DISPLAY-ről a „man X"-ben olvashatsz, ez esetben elégnek kell lennie annyinak, hogy van, és nem üres string.)

Perl-érdekességek (CPAN)

A Perlt Larry Wall – aki már előzőleg is ismert volt az rn és patch programok szerzőjeként – 1986-ban kezdte el írni, amikor rendszergazdaként úgy ítélte meg, hogy az előtte álló feladatokat egy új nyelv tervezésével tudná a legjobban megoldani. Larry nyelvésznek tanult, mert misszionáriusként az írott nyelv nélküli törzsek körében akarta a Bibliát terjeszteni. Amikor végül programozó lett, igyekezett olyan nyelvet létrehozni, ami az emberek gondolkodását és a természetes nyelveket imitálja. Fontos volt számára, hogy abban is hasonlítson műve a természetes nyelvekhez, hogy egy kezdő már a nyelv felületes ismeretével is hasznos programokat írhasson. Az egyik Perl-szlogen így hangzik: „Perl makes easy jobs easy and hard jobs possible" vagyis a Perlben a könnyű problémákat könnyű megoldani, a nehezeket pedig lehetséges. Szintén érdekes tulajdonsága a Perlnek, hogy még a legegyszerűbb dolgokat is sokféleképpen lehet megoldani. Mindenki kiélheti kreativitását, így a programozás sokszor szórakoztató játékká válik. Ezt a másik Perl-jelmondat fejezi ki: „There is more than one way to do it". A következő programsorok mindegyike ugyanazt az egyszerű feltételvizsgálatot végzi:

```
if(! $ertek) { print „hamis\n" }; unless($ertek) { print „hamis\n" }; $ertek or print „hamis\n"; !$ertek and print „hamis\n"; print „hamis\n" if (!$ertek); print „hamis\n" unless $ertek; $ertek ? undèf : print „hamis\n".
```

Vannak olyanok, akik szerint ez a sok lehetőség nem vezet jóra, mert így nehezebb a mások által írt programot megérteni. Valóban könnyű Perlben olvashatatlan programokat írni, ugyanakkor ez nem szükségszerű. A man perlstyle sokat segíthet e tekintetben. Található a mutatóbeli második címen egy érdekes írás Larry Wall tollából a természetes nyelvek és a Perl kapcsolatáról. A Perl akkor lett igazán népszerű, amikor a www berobbant az életünkbe. A HTML oldalak előállítása általában sok szövegfeldolgozást igényel, a Perlnek pedig ez a legnagyobb

erőssége. Azóta is a Perl a web motorja. A CGI programok 90 százalékát Perlben írják. Nélküle jóval nehezebb lenne számlálókat, vendégkönyveket, adatbázis-elérést és a sok egyéb szokásos (vagy szokatlan) szerveroldali programot elkészíteni. (A CGI a webszerver és a szerveren futó programok közötti adatcsere szabványa. Bármilyen nyelven lehet CGI programokat írni.)

A Perl felfogható úgy is, mint a klasszikus Unix szövegfeldolgozó eszközök (grep, sed, awk) kiterjesztett változata, ami a trükkös és hatékony shellszkriptektől, valamint a C nyelvből is kölcsönzött ötleteket. Történetében mérföldkő volt az 1994-ben megjelent 5-ös verzió, amelyben többek között a bonyolult adatszerkezetek építését megkönnyítő referenciákkal, és sajátos stílusban megvalósított objektumorientált lehetőségekkel bővült.

A CPAN (Comprehensive Perl Archive Network), az ingyenesen letölthető szkriptek és modulok hatalmas archívuma. Gyakorlatilag bármit megtalálsz itt, amire szükséged lehet, a kérdés csak az, hogy nem egyszerűbb e magadnak megírni, mint a más kódját megkeresni és megérteni.

Összehasonlítás az sh-val

A Perl szintén interpretált nyelv, de másképpen, mint a shell: míg a shell esetén a kód soronként kerül interpretálásra, a Perlben először az egész szkript „lefordul" egy belső formátumra és az kerül végrehajtásra. A gyakorlatban ez azt jelenti, hogy az esetleges szintaktikus hibák azonnal, indításkor kiderülnek. A shellszkriptek esetén előfordulhat, hogy elindul a program és futás közben áll le a szintaktikai hiba miatt.

Ráadásul így lehet optimalizálni is, így a Perl programok gyorsabbak, mint a shellhez hasonlóan soronként interpretált Tcl programok. A shellnél a Perl nemcsak az optimalizálás miatt gyorsabb, hanem azért is, mert a shell állandóan külső programokat hív meg, ezért az összegződő processzindítási idők lassítják.

Nézzük például, hogy hogyan írathatnánk ki a nevünket (nem a login, az igazit) egy shellszkriptben. Addig áramoltatjuk az adatfolyamot mindenféle szűrőkön keresztül, amíg nem marad más, mint amit kerestünk:

```
#!/bin/sh
juzername=$USER
igaziname=`cat /etc/passwd | grep „^${juzername}:" | cut
-d: -f5 | cut -d, -f1` echo A neved: $igaziname.
```

Egy processzt megtakaríthatunk, ha a catet elhagyjuk, és a file-t a grep argumentumának adjuk, de így is három processz marad. Ugyanez Perlben:

```
#!/usr/bin/perl -w
$file = „/etc/passwd";
# ha esetleg az USER környezeti változó
# nem lenne beállítva,
# akkor a LOGNAME -t is megnézzük.
$user = $ENV{USER} || $ENV{LOGNAME};
open(HANDLE, „< $file") or die „nem tudtam $file -t olvasásra megnyitni: $!\n";
while(<HANDLE>) {# soronként olvassuk a file-t
if (/^$user:/) {# ha a sor a $user-rel kezdődik
@mezok = split(':', $_);
# a: által elválasztott elemek ebbe a tömbbe kerülnek
$gcos = $mezok[4];
# a GCOS az ötödik elem
$gcos =~ s/^(.*?),.*/$1/g;
# nem kell, ami a vessző után van
print „A neved: $gcos\n";
}
}
close (HANDLE);
```

A getpwnam beépített függvény segítségével, jóval egyszerűbb lett volna a program, de úgy nem lett volna fair az összehasonlítás.

Adattípusok – reguláris kifejezések

A reguláris kifejezések egy külön nyelvet alkotnak az adott programon vagy programozási nyelven belül. Amikor egy regexpet megírsz, azt adod meg, hogy milyen követelményeknek kell megfelelnie egy stringnek. Minden elképzelhető követelményhez van (legalább) egy minta, amit a programon belüli kis interpreter balról-jobbra végighúz a beadott stringeken, míg végül megállapítja, hogy illeszkedett-e, és ha igen, akkor hogyan. A reguláris kifejezések sokszor ijesztően néznek ki, ami nem csoda, hiszen bonyolult algoritmusokat sűrítettek össze néhány karakterbe. Mégsem jut eszébe senkinek, hogy lemondjon róluk, mert még mindig egyszerűbb regexpeket komponálni, mint a nekik megfelelő algoritmust megírni. A reguláris kifejezések számos más unixos program (szövegszerkesztők, grep, sed, awk) hasznos részei, a Perl szövegfeldolgozó képességeihez is jelentősen hozzájárulnak. Használják őket grepszerű illeszkedés-vizsgálatokra: if($a =~ /minta/) {# $a illeszkedik a mintara}; sedszerű cserékre: $a =~ s/regiminta/uj/g. Továbbá egy stringnek az awk -F opciójának az általánosításával történő feldarabolására:

```
@feldarabolt=split(/minta/, $string)
```

A reguláris kifejezéseket nem szabad összetéveszteni a shell wildcardokkal. A shellekben * akármiből akármennyit jelent, a ? akármiből egyet. A reguláris kifejezésekben a . jelent akármiből egyet, a * az őt megelőző micsodából akármennyit, vagyis az „akármiből akármennyi"-t

a .* jelenti. A unixos programokat nagyjából két részre lehet osztani: a csak az alapvető reguláris kifejezéseket (.*^$) használókra (pl.: vi, grep, sed) és a kiterjesztett reguláris kifejezéseket ([m, n]+?|()) is használókra (egrep, awk, jerl). A perl reguláris kifejezéseit a perl man oldala írja le részletesen, íme egy nagyon rövid emlékeztető: ^ sor eleje, $ sor vége, \ következő karakter szó szerint, [] bármi, ami ezek között van, [^] bármi, ami nincs ezek között, + az előző egy vagy többször, ? az előző vagy van vagy nincs (0 vagy 1), | vagy, () csoportosítás, {n, m} az előző min. n-szer, de max. m-szer.

Reguláris kifejezések – esettanulmány

A Perl lehetőségeinek a bemutatására nézzünk egy problémát, amit nemrégiben maga a valóság szült: hogyan lehetne elérni, hogy egy stringben minden ismétlődő, nem alfanumerikus karakter cserélődjön le az adott karakterből egyre? Nem nehéz a probléma, a következő szubrutint bármelyik nyelven meg lehetne írni:

```
sub transform {$i = shift;
# $i a függvénynek adott argumentum
@chars = split( //, $i);
# karaktertömb a stringből
my $last; # lokális változók
my $o = „"; foreach $c (@chars) {# karakterenként nézzük
if($c =~ /\w/) {# ha alfanumerikus
$o = $o . $c; # hozzátesszük az $o string végére}
elsif($c ne $last)
{# ha nem az, de nem ismétlődik
$o = $o . $c; # akkor szintén
} $last = $c; }
return $o; # a szubrutin visszatérési értéke
}
```

Ezt – egy pillanatra elfeledkezve a man perlstyle átláthatóságra intő szavairól – néhány piszkos trükk felhasználásával akár egysorosra is átírhatnánk:

```
$output = join('', map { (/\w/) ? ( $_ ) : (($_ ne $last) ?
($_ && ($last = $_) ) : (($last = $_) && „") )} (split( //,
$input)));
```

(Ha valakinek ez a sor megtetszett, talál ennél profibbakat is a mutatóban említett címen.)

Azonban a tapasztalt Perlfelhasználó inkább rögtön egy reguláris kifejezést keres. A naiv megközelítés volna a következő:

s/(\W)+/$1/g. A \W illeszkedik minden nem-alfanumerikus karakterre, ezt zárójelbe tesszük, hogy a jobb oldalon lehessen hivatkozni rá (a $1 az első zárójel tartalmát jelöli), és egy +jelet biggyesztünk a végére, ezzel jelezve, hogy ha több mint egy lenne, akkor nekünk csak az első kell. Sajnos, ez nem pontosan azt végzi el, amit szeretnénk (regexpenél ez gyakran megesik), ha több különböző nem-alfanumerikus karakter jön, akkor azokból is csak egy marad, pedig mi azt szeretnénk, hogy csak az ismétlődő karakterek „hulljanak el". Egy lehetséges helyes megoldás a következő: s/(\W)\1*/$1/g.

Azt kell tudni, hogy míg a jobb oldalon $1-gyel hivatkozhatunk az első zárójel tartalmára (backreference), a bal oldalon \1 -gyel tehetjük ugyanezt, és most pont erre van szükségünk.

Python

Az 1991-es születésű Python nyelvet, a Monty Python csapatról nevezték el szerzői (a fő szerző: Guido Van Rossum). Ezt egy javított perlnek szánták, nagyjából ugyanazokra a feladatokra. Lelkes felhasználói azt jósolták, hogy rövidesen felváltja a Perlt minden területen (ez mindeddig nem következett be, és valószínűleg nem is fog). A javítások a „tiszta szintaxis"-ban, és abban jelentkeztek, hogy a Pythont már kezdettől fogva objektumorientáltra tervezték. A tiszta szintaxis a $, @ és % jelek mellőzésén kívül azt jelenti, hogy az utasításokat csakis bekezdésezés segítségével lehet csoportosítani (nincsenek zárójelek, nincs do/done, semmi). Lássunk egy példát!

```
#!/usr/bin/python
for i in range(10):
        if i > 3 and i < 7:
                print "se tul nagy se tul kicsi: ", i
```

Pythonban lehetetlen rosszul indentált programot írni, egyetlenegy felesleges space vagy tabulátor is szintaktikai hibát okozhat.

Érdekes, hogy a Pythonban hiányzik a lépegetős for, ha mégis szükség van rá, akkor a listán végigmenő for, és a listát generáló range függvény segítségével lehet emulálni.

Tcl/Tk és Expect

A TCL (Tool Command Language) nyelvet John Ousterhout 1988-ban kezdte el írni, amikor a kaliforniai Berkeley Egyetemen tanított. Szüksége volt egy olyan jól bővíthető interpretált nyelvre, ami lehetővé teszi, hogy az elkészült szkriptet bármilyen nyelvű programba be lehessen ágyazni. Hamarosan kiderült, hogy egy ilyen nyelvre másoknak is szüksége van, ezért az ingyenesen és forráskóddal hozzáférhető nyelv terjedni kezdett. A TCL végső formájában egy aránylag rövid parser rutinból és egy libraryből áll. Mindössze a parser rutint kell beépíte-

nünk programunkba, és máris használhatjuk a TCL parancsokat. Mivel a TCL igen jól hordozható, érdemes a program gerincét ebben írni, és csak a teljesítmény szempontjából kritikus részeket natív kódban implementálni. Tervezője is „ragasztó" programnyelvnek, azaz a kész modulok összeillesztésére szánta.

Ousterhout professzor szükségét érezte egy olyan eszközkészletnek is, amely lehetővé teszi grafikus felhasználói felületek gyors és egyszerű építését, így hát megírta a Tk-t (ToolKit) is (1989), ami gomboknak, scrollbaroknak, checkboxoknak stb. TCL-ből használható gyűjteménye. Ennek még nagyobb sikere volt, ezután sokan csak azért tanulták meg a TCL nyelvet, hogy a TK-t használhassák. Ez ma már nem annyira fontos, mivel azóta Perlből és Pythonból is el lehet érni a Tk-t, a PerlTk, illetve a tkinter csomagok segítségével.

A Tk-disztribúcióban található egy widget nevű program, ami a Tk lehetőségeinek egy demója. További magyarázgatás helyett arra buzdítok mindenkit, hogy próbálja ki ezt. A Tk-hoz van egy Tix nevű kiegészítés is, ami a Tk-elemekből felépített, bonyolultabb objektumokat tartalmazza (például dialógus file-kiválasztáshoz), és történetesen szintén létezik egy widget nevű demója.

1994–98 között John Ousterhout a Sunnál dolgozott. A Tcl/Tk közben eljutott a 8-as verziószámig, és portolták Windowsra, valamint Machintoshra is. 1998-ban Scriptics néven saját céget alapított, azóta a www.scriptics.com a Tcl/Tk-val kapcsolatos információk fő lelőhelye. A Tcl/Tk továbbra is ingyenes minden platformon. Pénzt csak a kiterjeszésekért, fejlesztőeszközökért stb. kérnek.

Expect

A Tk-n kívül a TCL-nek van egy másik érdekes kiterjesztése is, az Expect (Don Libes, 1990), ami egy általános megoldás a terminálról olvasó programok manipulálására. Megjegyzés: a CPAN-ból a Perlhez is letölthető egy Expect nevű modul, ami ugyanezt végzi. A telnet és az ftp perles automatizálásához még jobb, ha a Net::Telnet és Net::FTP modulokat (szintén CPAN) használjuk.

A Tcl/Tk egyik előnye, hogy sokféle felhasználóbarát grafikus „kütyüt" írtak hozzá, például a SpecTcl húzd és ejtsd felhasználói-felületépítőt, valamint a TclTutor interaktív tcl-tanítót.

A Tcl és a Tk programozása

A Tk-nak van egy wish nevű (WIndowing SHell) külön shellje, ami mindent tud, amit a tcl (mivel annak kiterjesztése), és ezen felül az említett Tk-widgeteket is kezeli.

Érdemes kipróbálni a következő szkriptet:

```
#!/usr/bin/wish
#ez egy kedves felhasználóbarát grafikus felület
button .b -text „Szia, te drága világ" -command exit
pack .b
```

A wish-t is lehet interaktívan használni. Ha beírod a parancssorba, hogy wish, előugrik egy üres ablak (végül is ez egy grafikus interpreter), az X terminálon pedig megjelenik egy százalékjel, ami után lehet a parancsokat írni.

Melyiket tanuljam?

A shellszkriptek írását mindenképpen érdemes megtanulni, legalább annyira, hogy egy „if-thenelif-then-fi"-t vagy egy for ciklust bármikor tudjunk rögtönözni. Az „igazi" (magasabb szintű) nyelvek közül első körben elég egyet kiválasztani: a Perl a rendszergazdák és webmesterek kedvence (ezen kívül ez a legelterjedtebb és ez rendelkezik a legjobb dokumentációval is), a Pythont inkább a komolyabb objekumorientált háttérrel rendelkező emberek kedvelik. A TCL mellett szól, hogy „anyanyelvi" szinten támogatja a Tk-t és az Expectet. Amit eddig láttunk, az csak étvágygerjesztő volt, az említett nyelvek, (különösen a Perl és a Python) sokkal többet tudnak: lehet bennük modularizáltan vagy objektumorientáltan programozni, adatbázisokat elérni, hálózatos programokat írni, (általában jeleskednek a nem túlságosan kompatibilis rendszerek „összeragasztásában"), és még sok hasznos dolgot elkészítésében. ■

MUTATÓ

http://www.ptug.org/sed/sedfaq.html
Larry Wall a Perlről:
http://www.wall.org/~larry/ natural.html
Magyar nyelvű online Perl dokumentációk:
http://www.infopen.hu/perl/
www.sch. bme.hu/misc/howto/perlhun.html
„Művészi" Perl programok:
work.media.mit.edu/tpj/contest-obfusc-entries
TCL/TK honlap: www.scriptics.com
Python honlap: http://www.python.org/
Expect honlap: http://expect.nist.gov/
Interaktív TCL oktatóprogram:
http://www.msen.com/~clif/TclTutor.html
TCL programgyűjtemény:
http://www.neosoft.com/tcl/contributed-software/
TCL faq:
http://www.tclfaq.wservice.com/tcl-faq/part1.html

Czakó Krisztián (slapic@chip.vogel.hu)

A Squid proxy cache beállítása

A web-szolgáltatás mellett a proxy-cache a Linux szerverek második legtöbbet használt szolgáltatása. Erre az egyik legjobb elérhető program a Squid, mely nem véletlenül az egyik legelterjedtebb proxy szerver a világon. Most ezt mutatjuk itt be.

Számos önkéntes fejlesztő is bedolgozott a Squid nevű IOC (Internet Object Cache) szerverbe, melyet Duane Wessels fejlesztett a National Laboratory for Applied Network Researchnél. (Az ANR-t a National Science Foundation alapította.) A program teljes egészében szabad szoftver, a GNU GPL2 alá tartozik. A fejlesztés az ARPA által alapított Harvest kutatási projekt „cached" programjából indult ki.

Mire jó, hol használható?

A Squid elterjedtségének okai között felsorolhatjuk, hogy számos operációs rendszeren fut (Linux, FreeBSD, NetBSD, BSDI, OSF és Digital Unix, Irix, SunOS/Solaris, NeXTStep, SCO Unix, Aix, HP-UX, valamint a GNU-Win32 csomaggal Windows NT alatt is) ingyenes és nagy teljesítményű FTP-, Gopher- és HTTP-s adatobjektum-cache. Nagy előnye, hogy a metaadatokat és az épp használt objektumokat a memóriában tartja, cache-eli a DNS (névszerver) kéréseket, támogatja a non-blocking DNS kéréseket, és implementálták benne a sikertelen kérések negatív cache-elését is. Támogatja az SSL-t (Secure Sockets Layer) és a teljes naplózást. Az ICP (Internet Cache Protocol) segítségével több proxy cache összekapcsolható, és további sávszélesség-takarékosságra alkalmazható. A proxykat egyenrangú és hierarchikus láncba kapcsolhatjuk.

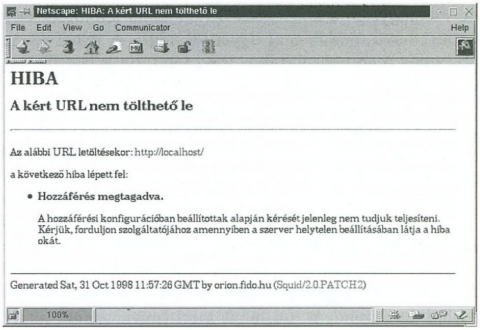
További hasznos funkció a „http accelerator" mód, melynek használatával a Squidet egy létező HTTP-szerver elé tesszük, ő szolgálja ki a klienseket, a tényleges HTTP-szervertől pedig csak ő kér adatot. A fentiek tükrében bizton állíthatjuk, hogy a Squid használható mindenhol, ahol az Internet-kapcsolatot egynél több felhasználó veszi igénybe, és előfordul, hogy azonos oldalakat hívnak le. A szolgáltató proxyjához hozzákapcsolva gyorsabbá és olcsóbbá tehetjük az Internet használatát. Szintén hasznos segítőtárs abban az esetben, ahol egy hálózat csak egy valós IP-számmal kapcsolódik az Internetre. Ebben az esetben ha csak FTP, Gopher és HTTP protokollra van szükségünk, IP-maszkolás helyett elég egy Squid is. Természetesen a kettő együtt is kiválóan működik.

Telepítés

Mivel a Squid legújabb verziója a legtöbb operációs rendszerhez előre csomagolt bináris formában is elérhető, forráskódja lefordítását itt nem ismertetjük. A program része a CD-n megtalálható Debian GNU/Linux 2.1-es rendszernek, azt a telepítésben leírtak alapján egyszerűen telepíthetjük (squid csomag). Telepítése után azonnal használható, konfigurálásához egyetlen (/etc/squid.conf) file szerkesztésére van szükség.

A kliensek konfigurálása

Proxynk kihasználására két lehetőség közül választhatunk. Az egyik megoldás általánosan használható: a kliensprogramokat (Netscape, Mosaic, Lynx, Arachne, Internet Explorer stb.) úgy állítjuk be, hogy a proxyt használják. Ezt böngészőtől függően manuálisan kell megtenni, illetve egyes böngészőknél, például



A Squid egy magyar nyelvű hibaüzenete

a Netscape-nél egy Javascript segítségével, ha a szolgáltató lehetővé teszi ezt számunkra. Az ehhez szükséges Javascript minta a CD-n megtalálható. A Squid beállításától függően más-más porton működhet, de az alapbeállítás szerinti portszám a 3128. Ezt nem szokás megváltoztatni, így ha nincs információnk, bízhatunk benne, hogy ott megtaláljuk. (Más proxyk gyakran a 8080-as portot használják.)

A másik – az operációs rendszertől erősen függő – lehetőség a transzparens proxy használata. Ez annyit jelent, hogy a kliensen semmit nem kell konfigurálni, a gépek által használt alapbeállítás szerinti átjáró (default gateway) automatikusan a megadott proxynak továbbít minden átmenő kérést. Ezt a lehetőséget az operációs rendszer kernelének támogatnia kell.

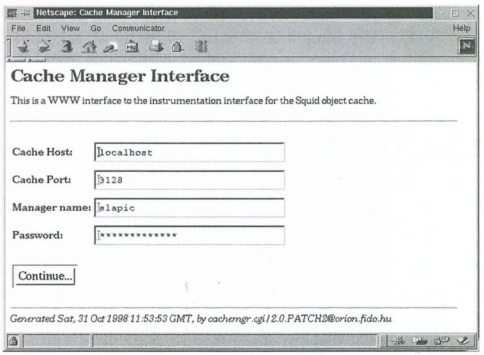
Alapvető beállítások

Mint már említettük, az alapszintű működéshez a Squid mellé csomagolt minta konfiguráció elegendő. Ennek ellenére érdemes átnézni azt, és behangolni a rendszerünkhöz. Elsősorban arra érdemes figyelni, hogy illetéktelenek ne használhassák a szerverünket –, ami ha fizetünk a letöltött adatok mennyisége után, pénzünkbe kerülhet. Ennek beállítására az ACL (Access Control List) opciókat kell használnunk. A Squid meglehetősen bonyolult ACL használatot tesz lehetővé, itt most csak az egyszerű eseteket tárgyaljuk. Felhasználóra és gépre vonatkozóan is korlátozhatjuk az elérést. Korlátozhatjuk a hozzáférést és a letölthető címeket. Először az általános hozzáférés beállításával foglalkozunk. Ha a felhasználók hozzáférését szeretnénk korlátozni, fel kell vennünk egy felhasználói név- és jelszó-adatbázist, valamint a felhasználónak be kell jelentkeznie a proxyra. A bejelentkezés egyszerű. A böngésző automatikusan megkérdezi az azonosítót és a jelszót, amint a proxyhoz kell fordulnia. Gépre vonatkozó korlátozásnál megadhatunk IP-számot, IP-tartományt, host- és domainnevet.

Hozzáférés korlátozása cím szerint

Ahhoz, hogy valakit korlátozni tudjunk, fel kell vennünk pár ACL nevet. Ezt az acl opcióval tehetjük meg. Az első listán négy acl sort találunk. Ezek jelentése a következő. A *mindenki* csoportba beletartozik mindenki. Ezt használjuk majd az általános tiltásra. A *manager* csoport a cache_object protokollt használók csoportja. Ők használhatják a Squid cache-managerét. A *localhost* csoport azokat tartalmazza, akik a helyi gépről jönnek, ezek lényegében mi vagyunk. A *belso* csoportba azon gépek tartoznak, melyek IP címe a 192.168.1.0-192.168.1.255 címtartományba esik.

A példa kedvéért tegyük fel, hogy ez a címtartomány a mi saját belső hálózatunké. Ezután következnek a http_access sorok. Ezekből az első a localhost kivételével mindenkinek megtiltja a cache-manager használatát (manager csoport), a második engedélyezi a belso csoportnak a cache használatát, a harmadik pedig mindenki másnak mindent megtilt. Az icp_access sorok hasonló módon engedélyezik az Internet Cache Protocol használatát a belső hálózatunknak, és megtiltja mindenki másnak. Figyeljük meg, hogy a Squid olvasási sorrendben értelmezi az access sorokat, azaz az első passzoló acl esetén megáll.



A Squid cache-manager kezelőfelületének bejelentkező képernyője

A http_access és az icp_access opció segítségével részletesen lehet beállítani a jogokat. Az *allow* és a *deny* paramétereket használhatjuk ehhez. Egyszerű esetekben erre nincs igazán szükség. Az acl opció a Squid proxy minden egyes funkcióját képes lefedni. A bejövő (src) és célcím (dst) mellett a Squid cache-manager hozzáférését is itt korlátozhatjuk, a címek mellett használhatunk domainnevet (srcdomain, dstdomain), a domainneveket átfogó regular expressiont (srcdomain_regex, dstdomain_regex), köthetjük a jogokat időhöz (time), URL-hez (url_regex, urlpath_regex), szolgáltatáshoz (port), protokollhoz (proto), böngészőhöz (browser), felhasználóhoz (user), adattovábbítási módhoz (method) – és még akad pár bonyolult lehetőség. Lehetőségeinknek csak a képzelőerőnk szab határt. Mivel e lehetőségek használatára csak ritkán van szükség, itt nem ismertetjük azokat részletesebben (a Squid angol nyelvű dokumentációjában le vannak írva a tudnivalók).

A nyelv beállítása

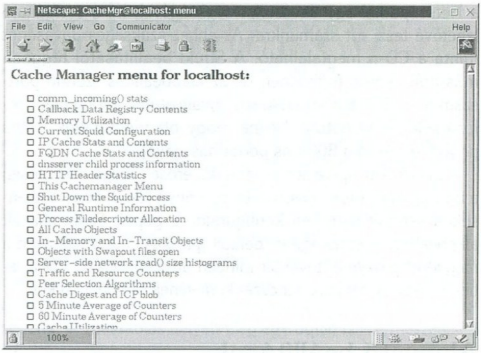
A 2.0 verzióban (és az utána következőkben) már lefordították magyarra az üzeneteket, melyeket a proxy akkor küld, ha valamit nem tud teljesíteni (például nincs hozzá jogunk, a szerver nem érhető el stb). Ahhoz, hogy az üzenetek magyarul jelenjenek meg, be kell állítani, honnan vegye azokat a Squid. A Debian Linuxnál a magyar nyelvű üzenetek a /usr/lib/squid/errors/Hungarian könyvtárban vannak. Ha ezt

beállítjuk az „error_directory /usr/lib/squid/errors/Hungarian" opcióval, máris magyarul „beszél" a cache. A megadott könyvtárban található file-okat szabadon átírhatjuk, ezzel is a saját igényeinkre szabva a rendszert.

Szomszéd proxyk használata

Ahhoz, hogy más proxy cache-hez csatlakozzunk, a *cache_peer* opciót használjuk. A proxyk kapcsolatában két fő megoldás lehetséges: apa-fiú (parent-child) és testvér (sibling) kapcsolat. Ha a társproxy nem ismeri az ICP protokollt, a mi proxynk nem tud intelligens módon kommunikálni vele, és ilyenkor csak az apa proxy beállítás működik.

A proxy hierarchiát úgy alakították ki, hogy azok a proxyk, melyek nagysebességű vonalon, vagy egy-egy hálózat kapcsolódási pontjain vannak, apa proxyként működjenek az alattuk levő gyerek proxyk számára, melyek testvér kapcsolatban vannak. A Squid proxy a kéréseket az alábbi módon próbálja kiszolgálni: ha a keresett objektumot megtalálja a saját cache-ében, átadja. Ha nem, végigkérdezi a testvér proxykat, akiktől csak akkor kapja meg az adatot, ha azok valamelyike megtalálja a saját cache-ében. Végül a szülő proxykhoz fordul, akiktől akkor is megkapja, ha azoknak nincs meg a cache-ükben, azaz az első szülőtől fogja megkapni, akinek megvan, vagy ha senkinek sincs meg, akkor attól, amelyik reagál. Itt leegyszerűsítve mutattuk be a működést. A társakkal történő kommunikációra az ICP protokollt használják a proxyk. Ha a saját proxynkban be szeretnénk állítani, hogy a proxy.provider.hu legyen a szülő proxyja, az alábbi sort kell beírni: „cache_peer proxy.provider.hu parent 3128 3130". Azaz a cache_peer opció után a proxy szerver neve (vagy IP-száma), azt követi a kapcsolat típusa (parent vagy sibling), végül a proxy és az ICP port. A sor végén további opciókat adhatunk meg. Ezek leírása a táblázatban olvasható. Itt is használhatjuk az acl-eket, hogy beállítsuk, mely kérések kerüljenek továbbításra a szomszédokhoz, melyeket töltse le közvetlenül a proxynk. Alapbeállításban minden objektumot előbb a testvérektől, aztán a szülőktől, végül közvetlenül kér le a Squid. Ha azt szeretnénk, hogy csak a helyi kérések menjenek közvetlenül, de azok mindig, minden mást csak szomszéd proxytól kérjen le, a második listán látható opciókat kell beírni. (A *mindenki* csoportnak az első listán definiált módon már léteznie kell.) Az első sor definiálja a helyi szervereket. Példánkban a vogel.hu domain és a localhost tartozik ide. Értelemszerűen a vogel.hu-t cserélje le a saját belső domainnevére, vagy a dstdomain helyett a dst paraméter használatával adjon meg IP-címtartományt. A második sor (always_direct) engedélyezi a helyi-szerverek csoportnak, hogy mindig közvetlenül töltse le az adatokat. A következő (never_direct) sor ugyanennek a csoportnak megtiltja, hogy ne közvetlenül szedje le az adatokat, végül az utolsó sorban a mindenki csoportnak engedélyezzük, hogy közvetlenül ne tölthesse le az adatokat.



A Squid főmenüje

Láthatóan ez némileg közvetett módszer, de így lehet a legtöbb megvalósítást lehetővé tenni. Ez a négy sor tömören

A kapcsolódó proxyk paraméterei

proxy-only	az innen letöltött adatot ne tároljuk.
weight=n	súlyozott szülő. Az *n* egy egész szám. Minél magasabb, annál jobban favorizáljuk az adott szülőt.
no-query	ne küldjünk ICP kéréseket ennek a szomszédnak.
default	az adott szülő használható mint utolsó lehetőség. Csak akkor van értelme, ha az ICP nem működik a szülőkkel.
round-robin	olyan szülők csoportjánál kell használni, akik nem ismerik az ICP-t, és körbe akarjuk őket kérdezni.
multicast-responder	az adott szomszéd egy multicast csoport tagja. Mi sosem küldünk neki ICP kéréseket, de a válaszokat elfogadjuk.
closest-only	az ICP_OP_MISS válaszokra csak CLOSEST_PARENT_MISS-t továbbítunk, és sohasem FIRST_PARENT_MISS-t.
no-digest	ne kérjünk cache digestet ettől a szomszédtól.
no-netdb-exchange	kikapcsolja az ICMP és RTT adatbázisok (NetDB) lekérését ettől a szomszédtól.
no-delay	kikapcsolja a „delay pool" beállítások késleltetését az adott szomszédra.

Egy acl minta

```
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl mindenki src 0.0.0.0/0.0.0.0
acl belso src 192.168.1.0/255.255.255.0

http_access deny manager mindenki !localhost
http_access allow belso
http_access deny mindenki

icp_access allow belso
icp_access deny mindenki
```

Kizárólag helyi kérések kiszolgálása

```
acl helyi-szerverek dstdomain vogel.hu localhost
always_direct allow helyi-szerverek
never_direct deny helyi-szerverek
never_direct allow mindenki
```

annyit jelent, hogy a vogel.hu és a localhost cím kivételével mindenhonnan csak a szomszéd proxykon keresztül töltjük le az adatokat. Akik ismerik a Squid 1.1-et, azok számára érdekes, hogy ez a módszer a local_domain, local_ip, inside_firewall és firewall_ip opciókat váltja le. További lehetőségünk, hogy meghatározzuk, egy-egy szomszédhoz mely domainek esetén fordulunk. Azaz megtehetjük, hogy a parent.foo.net szülőkhöz csak a .edu-ra végződő domainekbe irányuló kérés esetén fordulunk. Ehhez a következőt kell beállítani: „cache_peer_domain parent.foo.net .edu". Ez nem jelenti, hogy a .edu végű kéréseket csak oda továbbítja a proxynk, mindössze annyit jelent, hogy oda csak .edu végű kéréseket továbbít.

További opciók

A *cache_mem* opcióval állíthatjuk be a Squid által használt memória megközelítő értékét. Mivel egy-egy futó Squid processz nem csak a cache-re használ memóriát, ez nem jelenti a processz maximális méretét, az két–háromszor akkora is lehet. Ez csak a cache objektumok maximális memóriabeli méretét határozza meg. Alapbeállításban ez 8 Mbyte. Az ennek megfelelő sor: „cache_mem 8 MB".

A *maximum_object_size* határozza meg, hogy mekkora lehet az a legnagyobb objektum, amit a cache még elment lemezre. Az ennél nagyobb objektumokat (file-okat) nem tárolja. Alapbeállításban ez 4 Mbyte. Az ennek megfelelő sor: „maximum_object_size 4096 KB".

A *connect_timeout* opcióval megadhatjuk, mennyi idő elteltével adja fel a Squid egy adott URL letöltésének kísérletét. Alapbeállításban ez 120 másodperc. Az ennek megfelelő sor: „connect_timeout 120 seconds".

A cache-manager e-mail címét a *cache_mgr* opcióval adhatjuk meg. Ezt az e-mail címet fogja minden üzenete végére hozzáfűzni a Squid. Példa: „cache_mgr webmaster@vogel.hu". Ha a helyi kérések gyakran a domainnév megadása nélkül érkeznek, és ez gondot okoz, használjuk az „append_domain .vogel.hu" opciót. Ez a példa minden nem teljes névhez (nem teljes a név, ha nem tartalmaz pontot) hozzáfűzi a .vogel.hu végződést. Értelemszerűen állítsuk be a saját domainünkre. A *http_anonymizer* opció lehetővé teszi, hogy a célszerver elől elrejtsük a minket használó kliensek számos adatát. Ezzel a saját belső információinkat is védhetjük. Három lehetséges beállítás van: off, azaz nincs védelem; standard, azaz csak pár specifikus adat nem juthat ki és paranoid, azaz csak pár specifikus adat juthat ki. Ha emellett a *fake_user_agent* opciót is alkalmazzuk, az utóbbi még a böngésző tényleges nevét is „meghamisítja", azaz megadhatjuk, mit lásson böngészőnek minden szerver, ahova kapcsolódunk. Például egy teljesen fals adat: „fake_user_agent Nutscrape/1.0 (CP/M; 8-bit)".

További lehetőségek

A fent ismertetett opciók és lehetőségek csak egy töredékét képezik a Squid teljes tudásának. Egyrészt az acl lehetőségei sokkal szélesebbek az itt felsoroltaknál, másrészt számos olyan opciót nem is említettünk, melyek hatékonyan használhatók egy hálózat forgalmának szabályozásában. Erre példa a delay pool opciók használata, melyekkel korlátozhatjuk egy-egy kliens vagy hálózat maximális forgalmát, a teljes HTTP, illetve FTP forgalmat, de ezek beállításához már a Squid működési elveinek alaposabb ismerete szükséges.

Más protokollok

Mint látható, a Squid csak a HTTP-, Gopher- és FTP-kapcsolatokhoz működik, igaz, ott kiváló szolgáltatásokat nyújt. Az FTP esetében ez annyiban korlátozott, hogy csak URL-lel megadott letöltéseket képes elvégezni, azaz egy hagyományos FTP protokollt használó klienshez nem jó. Ezekhez, és a többi protokollhoz, ha nem elégszünk meg az IP-maszkolással (mely a Network Address Translation, azaz NAT egy speciális esete), vagy ha a rendszerünket tűzfallal akarjuk védeni, és le kívánjuk tiltani az átmenő forgalmat, más proxykat kell használnunk. E feladatokra a Socks és Delegate szoftverek valók. Az előbbi egy részben pénzes (magáncélra ingyenes), az utóbbi egy GPL licensz alá eső szabad szoftver. ■

Szieberth Dénes
dino@iris.inc.
bme.hu

A pingvin és a kémia

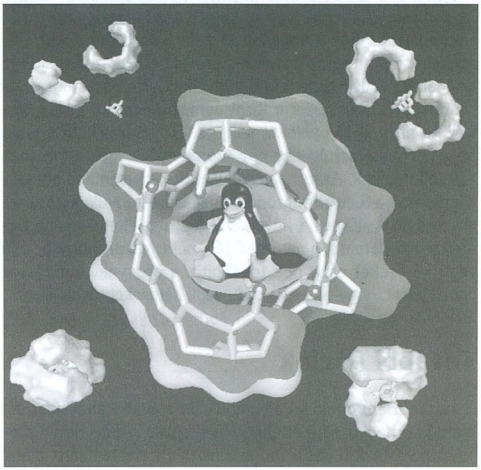
Miközben a hálózati szerverek, irodai és egyéb „köznapi" alkalmazások piacán még éppen hogy elkezdődött a Linux térnyerése, egyes kevésbé közismert területeken már most óriási szerepet tölt be az ingyenes Unix-klón. E területek közé tartozik egy az utóbbi években rendkívül dinamikusan fejlődő tudományág, a molekulamodellezés.

E tudományterület ugyanannak a ténynek köszönheti felvirágzását, ami a Linux születését és elterjedését is lehetővé tette. A gyors, és olcsó számítógépek elterjedésével jóformán mindenki számára hozzáférhetővé váltak. S hogy az operációs rendszerek széles skálájából miért pont a Linux lett az ezen a területen dolgozó kutatók egyik kedvenc munkaeszköze? Ha összevetjük a molekulamodellezés igényeit a Linux tulajdonságaival, egyértelművé válik a válasz.

Számítógépek használata a kémiában

A kvantumkémia, illetve a kvantumkémiára épülő molekulamodellezés a számítástechnika egyik legintenzívebb felhasználója. Nem csoda: e tudományág a számítástechnikai háttér nélkül jóformán nem is életképes. Noha elméleti alapjait már jóval korábban kidolgozták, a szűkösen elérhető számítási teljesítmény miatt csak a legutóbbi évtizedben lépett ki az elméleti kémikusok, mint felhasználói kör kezei közül, s villantott fel olyan, az ipar számára is értékes lehetőségeket, hogy a nagyobb vegyi- és gyógyszergyárak is érdemesnek tartották saját, molekulamodellezéssel foglalkozó kutatók alkalmazását.

Habár a molekulamodellezés szó hallatán legtöbbünknek pálcikákkal összekapcsolt színes golyócskákkal telizsúfolt képernyő jutna eszünkbe, a szükséges számítástechnikai háttérnek csupán egyik, nem is a legfontosabb eleme a grafikai prezentációs-képesség. Alapvető szükséglet a lebegőpontos számokkal való gyors műveletvégzés, a nagyméretű memória, valamint a gyors és nagyméretű háttértár. Egy



példán keresztül érzékeltetve: egy néhányszor tíz atomból álló molekula tulajdonságainak kvantumkémiai számításokkal történő meghatározása egy mai csúcskategóriájú PC-nek (Pentium II 450 MHz, 512 MB RAM) a számítások pontosságától függően több napját, esetleg hetét is igénybe veheti, közben 5–10 Gigabytenyi merevlemezt lefoglalva. Ilyen óriási számítási teljesítményigényt korábban csak a nagyobb

Molekulamodellezés alatt itt kémiai rendszerek tulajdonságainak számítástechnikai eszközökkel történő, legtöbbször a kvantumkémia elvein alapuló meghatározását értjük.

Az 1998-as kémiai Nobel-díjat e tudomány képviselői kapták Walter Kohn és John A. Pople.

egyetemek szuperszámítógépei, kicsit később, pedig a sokkal alacsonyabb áruk miatt nagyobb számban vásárolható munkaállomás kategóriájú gépek voltak képesek kiszolgálni.

Az utóbbi egy–két évben azonban a PC-k sebessége megközelítette az egykori munkaállomás kategóriáét, s egyre több felhasználó találja gazdaságosabbnak a PC-k alkalmazását a nagyobb teljesítményt igénylő területeken is. Azt azonban több tényező is gátolta, hogy a molekulamodellezésben is elterjedjenek a PC-k. A kvantumkémiai programok döntő többsége Unixra íródott, s a legtöbbször DOS/Windowst futtató PC-kre nehezen lehetett csak átvinni őket. A Linux megjelenésével e programok forráskódjai kevés átalakítással, vagy átalakítás nélkül alkalmazhatóvá váltak PC-ken, s az egyébként is Unixhoz szokott felhasználóknak sem kellett nagy energiát befektetni egy új operációs rendszer megtanulásába.

A Linux alatt futó programok általában gyorsabbak voltak, mint DOS/windowsos társaik, és számos olyan lehetőséget is kihasználtak, amire a többi PC-s operációs rendszer nem fordított elég figyelmet. Például a több felhasználó és feladat párhuzamos, illetve egymást kevéssé befolyásoló kiszolgálása, a diszkműveletek gyorsítása (software-RAID) stb. Ilyenek még az Interneten keresztül történő könnyű elérhetőség (gyakran előfordul, hogy például egy Japánban tartózkodó kutató, valahol Európában található számítógépen akar néhány számítást lefuttatni), illetve a feladatok időbeni elosztását segítő queue-rendszerek.

A Linuxot vonzóvá tette stabilitása is (Kellemetlen dolog, ha egy több hétig tartó számítás befejeződése előtt néhány órával lefagy a gép.). A Linux lehetőséget adott arra is, hogy a nagyobb számítási teljesítményt több PC összekapcsolásával érhessék el (pl.: BEOWULF jellegű clusterek). Végül, de nem utolsósorban: a Linux ingyenes szoftver, ami egyáltalán nem elhanyagolható a molekulamodellezéssel foglalkozó csoportok általában elég népes gépparkja esetén.

Mit hozhat a jövő?

Egyes előrejelzések szerint a közeljövőben a Linux–PC párosítás sok hagyományos területéről ki fogja szorítani a gyengébb ár/teljesítmény aránnyal rendelkező unixos munkaállomásokat. Ezek gyártói több irányban is próbálnak szabadulni a szorításból: van aki leáll a processzorok fejlesztésével, és Intelre alapozva képzeli el a jövőt. Némelyek a szoftverfejlesztés költségeit csökkentik és a Linux adaptálásával próbálják hardvereiket vonzóbbá tenni, és persze akadnak, akik figyelmüket inkább egy, a PC-k által még nem komolyan fenyegetett területen, a multiprocesszoros rendszerek felé fordítják.

Akárhogyan is alakul azonban a munkaállomások piaca, a következő egy–két év a Linux évének ígérkezik a molekulamodellezésben. ■

Bujtár János
bujtar@mail.de-dasz.hu

Linux az energia-szolgáltatásban

Bizonyára sokan kíváncsiak arra, hol használnak manapság Linuxot. A világon egyre többször halljuk, hogy komoly, nagy cégek alkalmaznak sikerrel linuxos rendszereket, de Magyarországról kevés ilyen információ jut el a sajtóhoz. Az alábbiakban közreadunk egy ilyen sikertörténetet, és várjuk olvasóink hasonló történeteit.

A bemutatott cég – a Dél-dunántúli Áramszolgáltató Rt. – területileg a Nagykanizsa, Balaton, Duna és a déli országhatár által közrefogott területen teremti meg a villamosenegria-ellátást a háztartások és üzleti ügyfelek számára.

A részvénytársaság központja Pécsett található. Itt üzemel az IBM S/390 típusú mainframe-gép. Ezen fut az integrált gazdasági és az értékesítési tevékenységet támogató szoftverrendszer. A mainframe gépre WAN-hálózaton kapcsolódik több mint ezer PC-s munkaállomás. Ezért elengedhetetlenül fontos az előbb említett nagygépes rendszerek elérhetősége. Emellett a gépek egy részén tipikus irodai alkalmazások futnak.

A Linux első alkalmazása

A történet kezdete két évvel ezelőttre nyúlik viszsza, amikor az Internet egyre nagyobb magyarországi térhódítása következtében úgy döntöttünk, mi is rákapcsolódunk az információs szupersztrádára. Egy–két hónapos tesztelés után döntöttünk úgy, hogy a bérelt vonalas 64 Kb/s elérés lenne számunkra a legmegfelelőbb. Az első lépéseket Microsoft platformon tettük meg.

A szerver egy Windows NT 4.0 volt. Használtuk az IIS-t és egy SMTP/POP3/IMAPI4 levelező szoftvert. Ez a gép szolgálta ki a DNS-, proxyszerviz funkciót is.

Hetente, de legjobb esetben kéthetente újra kellett indítani a Windows NT-t, mert egyszerűen elfogyott a memória. És akkor történt egy nagyon fontos dolog: az Interneten böngészve meglátogattuk az egyik társáramszolgáltató weboldalait. Kiderült, hogy amit mi Pentium 100 MHz gépen 64 MB RAM-mal, 2 GB merevlemezzel próbálunk meg használni Windows NT alatt, azt ők 486 DX/2 66 MHz, 16 MB RAM gépen bonyolítják az ingyenes Linuxszal. Úgy döntöttünk, hogy nincs vesztenivalónk, érdemes foglalkozni a Linuxszal. Debiannal kezdtük, és bár azóta több disztribúció is megfordult nálunk, biztosíthatunk mindenkit, hogy lényegi különbségek nincsenek. Elkezdődött a Linux „megszelídítése".

Először a levelezést tettük át Linuxra. Ugyanaz a rendszer (Sendmail), amit anno 1997 augusztusában a Linux levelezőlista segítségével beüzemeltünk, a mai napig tökéletesen működik. A konfiguráció a következő: no-name alaplap, 486 DX4/133 MHz, 32 MB RAM, 1 GB HDD és a körítés. Csak akkor kell a géphez nyúlni, amikor új e-mail címet kell felvenni vagy frissíteni kell a rendszer valamelyik részét. Újraindításra három alkalommal került sor: egyszer a VGA-kártya adta fel a harcot, kétszer kernel frissítés után.

Következett a proxy-szerver leváltása. Egy 486 DX2/66 MHz gépre „felhúztunk" egy Linuxot és a Socks-5 proxyt. Tökéletesen működött vele a Netscape, az Internet Explorer az IRC és a többi alkalmazás is. Egyszerűen konfigurálható, a naplófile-ok készítése pedig gyerekjáték volt.

Közben meggyűlt a bajunk az NT-s DNS-szerver szolgáltatásaival. Természetesen a következő lépés az volt, hogy áttettük Linuxra. Azóta nincs gond vele.

A webszerver áttelepítése

Az Apache kitűnő munkát végez. Robosztus, jól konfigurálható, biztos megoldás. Voltak és vannak alkalmazásaink, amelyek a socksot nem támogatják. Ezek miatt váltottunk a Squid Proxy szerverre. Még több, részletesebb naplózási lehetőség, a forgalom körülbelül 30 százaléka a Squid cacheből jön (ha a forgalom utáni díjat számlázzák, egyáltalán nem mindegy), tartalomszűrési lehetőség és még sorolhatnánk az előnyeit. A Win95-ös kliens gépeken a Netscapeben egyedül az automatikus proxy-konfigurációt kellett beállítani, és máris mindenkinek olyan szolgáltatást tudtunk nyújtani, amire jogosult.

Megadott IP-címekről csak a kívánt időben csak a kívánt szervereket érhetik el a felhasználók. Ha bármilyen változás történik az Internetet kiszolgáló szerverekben, akkor egyetlen helyről elintézhető az adminisztráció. Nagyon fontos dolog ez, ha figyelembe vesszük, hogy 300–500 gép hozzáféréséről van szó. Időközben egy intranetes webszolgáltatást is beindítottunk ugyanazon a gépen, amelyik a DNS-, FTP- és proxy-szervert is ellátja. Minden éjszaka a tehermentes órákban a szerver feldolgozza a naplófile-ok tartalmát, és mentést végez.

Legutóbb komolyan fontolóra vettük a CD-jogtár használatát Linux alól. A szoftver szerzőinek köszönhetően – támogatást nyújtanak Linuxra – megoldottnak tűnik a program használata WAN-hálózaton keresztül lassabb (64 Kb/s) vonalakon is. Ilyen módon elég egy helyen tárolni, igény szerint bárki elérheti, s a CD-verziók követése és adminisztrálása egyszerűbbé válik.

Nyomtatószerver

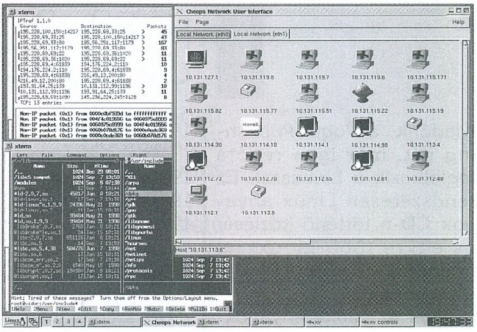
Cégünk nagy részének file és nyomtató kiszolgálását eddig Novell Netware szerverek alatt végeztük. Az egyik ilyen gépre – a nagy leterheltség miatt – a maximálisan lehetséges 100 felhasználóból szinte állandóan 95–100 van bejelentkezve. A megoldás a Mars lett. A 250 kliensre konfigurált Netware-emulátor feladata a többi szerver tehermentesítése volt. Ez idáig tökéletesen ellátta ezt a feladatot, csak az volt a „baj" vele, hogy az éppen induló kliensek kéréseire a Mars válaszolt legelőször, így akinek nem volt beállítva a preferált szerver, a Marsra próbált meg bejelentkezni a saját szervere helyett. Ez a hiba persze a kliensek beállításával kiküszöbölhető.

Megoldás a mentésre

Ezt a feladatot is „véletlenül" Linuxon oldottuk meg: kineveztünk egy gépet, amelyik az éjszakai órákban összegyűjti a Netware hálózatról a mentésre szánt anyagokat és archiválja őket.

A hálózati eszközök felügyelete mindig is kiemelt feladat volt cégünknél. Több mint 1000 gép és közel 100 router/gateway/hub koordinálását és üzemeltetését egy Sun Sparcstation 250-ről működtettük Solaris alól a HP OpenView segítségével.

Többszöri rendszerösszeomlás után – mivel a hardverbővítés és a szoftverfrissítés akkora összeget emésztett volna fel, amit mi már nem kívántunk megadni értük – a Linux mellett döntöttünk. Most Red Hat Linux alatt kiválóan érzi magát a Sun, a Gnome pedig kellemes felületet ad a Scotty/Tkined nevű hálózatfelügyelő program használatához.



Hálózati felügyelet Linux alól

Azokat a feladatokat, amelyeket pedig a Tkineddel nem lehet elvégezni, saját fejlesztésű szkriptekkel oldjuk meg.

A legutolsó projektünk feladata annak a bizonyítása, mennyire megoldható a nagyszámú (100–300) kliens gép remote-boot módszerrel történő indítása. Ez egy több száz, esetleg több ezer gépes hálózatnál nagyon kényes és komoly feladat. Ha megoldható a teljes rendszer felügyelete, szoftver installálása, vírusmentes környezet biztosítása, akkor nagyon megkönnyítené a rendszeradminisztrációt.

Az elképzelésünk a következő: minden egyes hálózatba kapcsolt gép az Ethernet-csatolón található booteprom segítségével kér egy IP-címet a hozzá legközelebb eső DHCP-szervertől. A felhasználó név és jelszó megadása után választhat, milyen operációs rendszert kíván használni, DOS/Win 3.1-et, Win95-öt vagy Linuxot.

A szerverről a kiválasztott rendszer egy előzetes gyors merevlemez-formázás után betöltődik a hálózaton keresztül, és a felhasználó NFS vagy Sambán keresztül eléri a file-okat és nyomtatókat. Ez az elv biztosítja, hogy mindenki csak azokat az erőforrásokat és programokat érje el, amire a rendszergazda jogot adott. Majdnem száz százalékosan kiküszöbölhetők a vírusok (a makrovírusok kivételével), és a programok frissítése is nagyon egyszerű. Egyelőre igen biztatók az eredmények, ha a terv megvalósul, beszámolunk róla.

Röviden ennyit tartottunk érdemesnek elmondani arról, hogy milyen módon térünk át fokozatosan a Linuxra. Persze az út nem volt mindig olyan, mint a fáklyásmenet, voltak nehéz helyzetek, de köszönet mindazoknak, akik segítették munkánkat: főnökeinknek – akik nem kötötték ki, milyen rendszer segítségével kell megoldani az adott problémát, hanem a lényeg az volt, hogy jól és megbízhatóan működjön –, valamint a magyar linuxos társadalomnak, akik minden nap nagyon sok segítséget és bátorítást nyújtottak.

Mindenkinek javasoljuk, érdemes kipróbálni, ígérjük, senki sem fogja megbánni a használatát. Bármilyen kérdést, észrevételt szívesen veszünk! ■

Bibliográfia

Kiadónk, a Vogel Publishing Kft. már igen régóta foglalkozik a Linuxszal. Ez rendszeresen Linux anyagok CD-n történő közlését és egyre több Linuxszal foglalkozó cikk publikálását jelenti.

Mivel az eddigi anyagokról sehol nem jelent meg egy vázlatos összefoglaló, itt most összegyűjtöttük az 1997. óta megjelent cikkeket, valamint a CD-kre felkerült disztribúciókat. Az alábbiakban a CHIP Magazin egyes számaira lebontva látható, hogy abban milyen Linuxszal foglalkozó írás jelent meg. Alatta található azon CD-k megjelenési dátuma, melyeken Linux-disztribúciót adtunk közre. Természetesen a CHIP Magazin CD-mellékletein minden hónapban 100–200 megabyte Linux anyag is található, ahol az aktuális disztribúciók frissítései, kiegészítői szerepelnek, így a mostani Debian GNU/Linux 2.1 későbbi frissítései is ott fognak helyet kapni.

Az említett CHIP Magazin számai általában megvásárolhatók a kiadónál (Budapest, XIII. Hajdú utca 42–44., telefon: 349-4768). Kérjük, hogy a régebbi cikkek olvasásakor vegyék figyelembe a Linux rendszer és a linuxos programok gyors fejlődését, ami jelentheti egy-egy tény vagy gyakorlati megoldás elavultságát.

1999. május	188. oldal	IP-láncok 2. rész (IP-maszkolás, számlázás)
1999. április	154. oldal	IP-láncok 1. rész (IP-tűzfal)
1998. december	171. oldal	GNU proxyval (Squid)
1998. november	149. oldal	Kommunikációs szerver I. (proxy, IP-maszkolás)
1998. október	173. oldal	Levelezés Linux alatt (smail, ppp, pop3, imap, procmail)
1998. szeptember	122. oldal	Kernelmodulok (használat, beállítás)
1998. augusztus	104. oldal	Az XFree86 beállítása
1998. július	104. oldal	Új mag ültetése Linux alatt (Kernel 2.0 konfigurálás, telepítés)
1998. június	120. oldal	Beszéljünk magyarul! (ékezetesítés, magyar billentyűzet, konzol, X11, TeX, LyX)
1998. május	167. oldal	Csomagkezelés (RedHat, rpm)
1998. április	125. oldal	Kezdő lépések (RedHat, floppy- és CD-használat, autofs)
1998. március	18. oldal	Új Linux (RedHat 5.0 telepítés)
1998. február	78. oldal	Linux-járóka (Debian 1.3 telepítés, Internet, diald, dpkg)
1998. január	76. oldal	Linuxban otthon van (RedHat 5.0, Applixware 4.3.7)
1997. december	92. oldal	A kényelmes Linux (K Desktop Environment)
1997. szeptember	98. oldal	Vörös kalapban táncolok (RedHat-telepítés)
1997. július	66. oldal	Linuxos iroda (Applixware)
1997. május	110. oldal	A Linux pókfonala (Internet, PPP, RedHat 4.1)
1997. február	60. oldal	Mire jó a Linux? (A Linux általános bemutatása)

CD-k:

1998. október	S.u.S.E. Linux 5.3
1998. március	Red Hat Linux 5.0
1997. augusztus	CHIPTár Linux: Debian GNU/Linux 1.3.1
1997. március	Red Hat Linux 4.1
1996. november	Red Hat Linux 4.0

1997. augusztusában jelent meg a 9-es számú linuxos CHIPtár. Ennek CD-mellékletén a Debian GNU/Linux 1.3.1-es verziója található meg, az újságban pedig érdekes ismertetők és linuxos megoldások olvashatók.

Slapic

Tisztelt Olvasó!

Szeretnénk megkérni Önt, hogy segítse leendő kiadványaink szerkesztését azzal, hogy kitölti az alábbi kérdőívet. Vágja ki és faxon vagy összehajtva levélként küldje vissza szerkesztőségünkbe. Válaszai által pontosabb képet kaphatunk olvasóink érdeklődési területeiről, és arról, hogy milyen információkat keresnek, illetve mit várnak tőlünk.
A kérdőívet visszaküldő olvasóink között a Pilátus-Comp Kft. jóvoltából magyarított Linux Office Suite 99, S.u.S.E. és RedHat Linux termékeket sorsolunk ki.

1. Mióta foglalkozik a Linuxszal?
- ☐ kevesebb, mint 6 hónapja
- ☐ fél–egy éve
- ☐ 1–2 éve
- ☐ 2–4 éve
- ☐ több mint 4 éve

2. Hol találkozott először a Linuxszal?
- ☐ Egyetemen, főiskolán
- ☐ Középiskolában
- ☐ Más oktatási formában (például tanfolyamon)
- ☐ Munkahelyén
- ☐ Barátjánál, ismerősénél
- ☐ Kiállításon, szakvásáron

3. Hol szerezte be első Linuxát?
- ☐ CHIP Magazin CD-mellékletén
- ☐ A CHIPtár Linux kiadványban (az 1997-es vagy jelen kiadványban)
- ☐ Másik újság CD-mellékletén: .....
- ☐ Boltban vásárolt dobozos termékben
- ☐ Internetről töltötte le Ön, vagy Önnek valaki más
- ☐ Ingyen osztott bemutató CD-n

4. Mivel foglalkozik Ön?
- ☐ Tanuló
- ☐ Rendszergazda
- ☐ Programozó

Döntéshozó:
- ☐ tulajdonos
- ☐ elnök
- ☐ ügyvezető igazgató
- ☐ kereskedelmi igazgató
- ☐ gazdasági igazgató
- ☐ egyéb, mégpedig .....

- ☐ Vállakozó
- ☐ Egyéb

5. Ha dolgozik, munkahelyén van-e befolyása a szoftverbeszerzésekre?
- ☐ Ön dönt
- ☐ Részt vesz döntéselőkészítésben
- ☐ Nincs módja befolyásolni a döntést

6. Milyen írásokat olvasna legszívesebben egy Linuxszal foglalkozó újságban?

Kérjük rangsorolja az alábbiakat, 1-gyel jelölve a legfontosabbat:
- ☐ Linux híreket, újdonságokat
- ☐ Konkrét, Linuxot alkalmazó vállalati rendszerek ismertetését (nem gyakorlat jelleggel)
- ☐ Programok ismertetését (nem gyakorlati jelleggel)
- ☐ Kezdő gyakorlati leírásokat
- ☐ Haladó gyakorlati leírásokat
- ☐ Profi gyakorlati leírásokat

7. Milyen céllal használ Ön Linuxot?
- ☐ Munkaeszközként
- ☐ Hobbi célra

- ☐ Csak ismerkedik vele
- ☐ Iskolában tanulja, de máshol még nem használja

8. Mit szeretne látni egy Linuxszal foglalkozó újság CD-mellékletén?

- ☐ Teljes Linux disztribúciót
- ☐ Nagyobb mennyiségű, disztribúcióhoz csomagolt programot
- ☐ Nagyobb mennyiségű forráskódban leközölt programot

9. Melyik Linux disztribúciót látná legszívesebben CD-mellékletünkön?

- ☐ Debian GNU/Linux
- ☐ Red Hat Linux
- ☐ S.u.S.E. Linux
- ☐ Slackware
- ☐ Egyéb: ....

10. Használ-e másik operációs rendszert a Linux mellett?

- ☐ Nem
- ☐ Igen, Windows 95/98-at
- ☐ Igen, Windows NT-t
- ☐ Igen, OS/2-t
- ☐ Igen, másik Unixot
- ☐ Igen, DOS-t (Windows 3.1-et)
- ☐ Igen, éspedig: ....

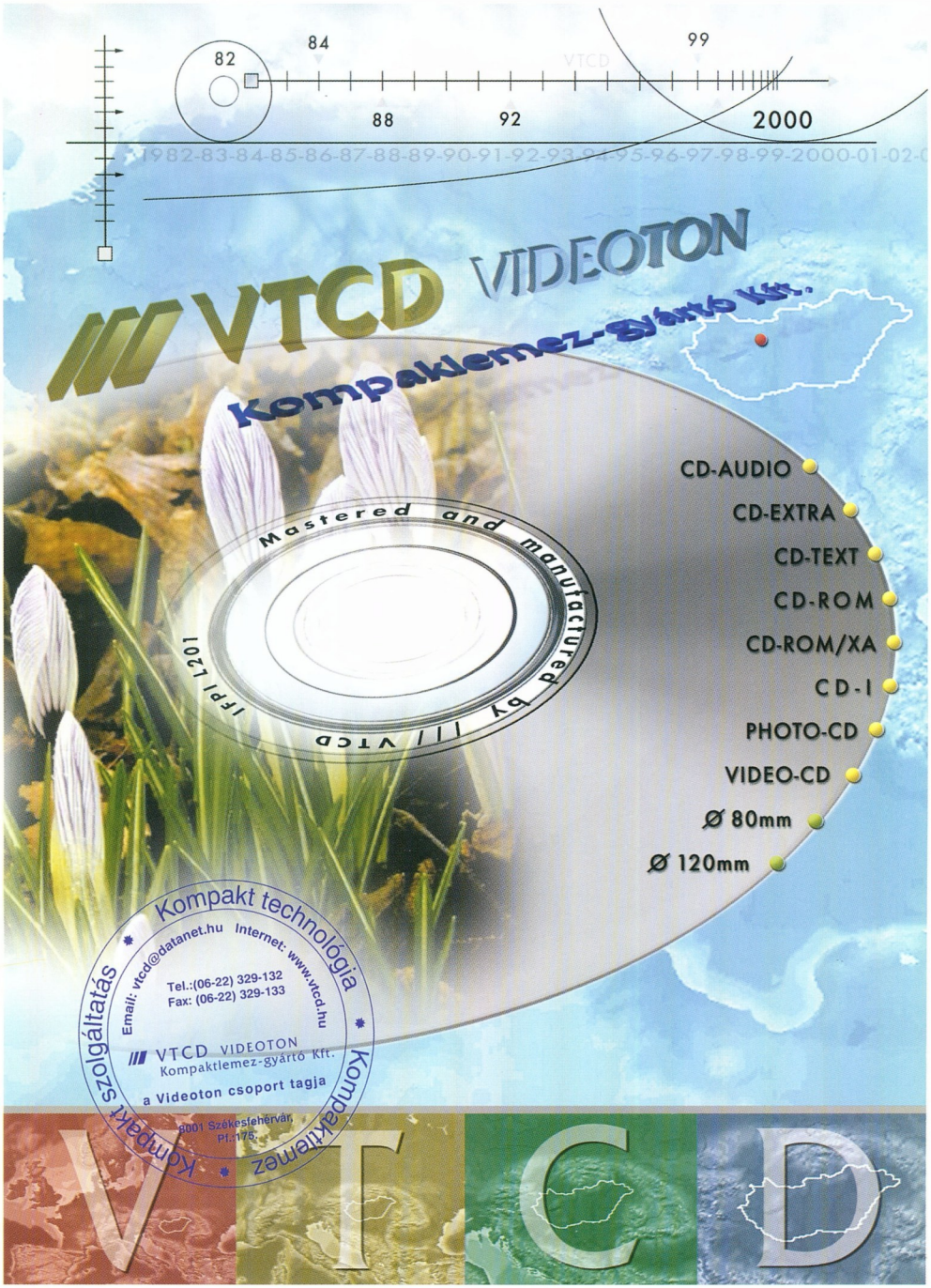
11. Mire használja a Linuxot?

- ☐ Intranetes szerverként
- ☐ Internetes szerverként
- ☐ Irodai alkalmazások futtatására
- ☐ Programfejlesztésre
- ☐ Játékra
- ☐ Multimédiára
- ☐ Egyéb

Feladó:

VOGEL Publishing Kiadó Kft.
Budapest 3.
Pf. 210
1300

Belföldre bér-mentesítés nélkül feladható, az esedékes díjakat a címzett fizeti


82
84
88
92
99
2000
VTCD
VIDEOTON
Kompaktlemez-Gyártó Kft.
Mastered and manufactured by /// VTCD
CD-AUDIO
CD-EXTRA
CD-TEXT
CD-ROM
CD-ROM/XA
CD-I
PHOTO-CD
VIDEO-CD
Ø 80mm
Ø 120mm
Kompakt technológia
Kompaktlemez szolgáltatás
Email: vtcd@datanet.hu
Internet: www.vtcd.hu
Tel.:(06-22) 329-132
Fax: (06-22) 329-133
VTCD VIDEOTON
Kompaktlemez-gyártó Kft.
a Videoton csoport tagja
8001 Székesfehérvár,
Pf.175.
V
T
C
D