Gumimatraccal vagy légykukaccal indítsuk a Linuxot?

A címben olvasható magyarításért egyszer talán nívódíjat kapok, de addig is az eredeti angol elnevezéseket használom írásomban, melyben a GRUB rendszerindítót mutatom be.

Legtöbbünk találkozott már a LILO-val, amely minden rendszerindításkor arról gondoskodik, hogy a Linux elinduljon. Neve nem az angol lilo (felfújható gumimatrac) szóból ered, ahogyan azt a magyar olvasó első pillantásra gondolhatná, hanem a LInux LOader szavak összevonásával a Unixban megszokott módon kiagyalt mozaikszó, melynek jelentése: *Linux-betöltő*. A GRUB a másik számos jó tulajdonsággal bíró rendszerindító, amelyet *Erich Boleyn*, *Gordon Matzigkeit* és *Okuji Yoshinori* fejleszt.

A GRUB egy mozaikszót takar, melyet a GRand Unified Bootloader szavakból raktak össze és ekként magyarítható: Nagy Egyesített Rendszerindító. A szerzőket nyilván nem zavarta, hogy a született angol nyelvű felhasználó a szó hallatán lárva, légykukac vagy egyéb dögféregcsali típusú képzettársításokra ragadtathatja magát. Ha rendszerindítóként a GNU GRUB-ot választjuk, a számítógép bekapcsolása után átveszi a vezérlést a BIOS-tól, majd betölti és felélesztí a rendszermagot. Ez utóbbi felel a rendszer többi részének indításáért. A GRUB egyelőre szorosan kötődik a PC-hez, de tervezik, hogy később másféle számítógép-kiépítésre is átviszik. PC-n viszont számos operációs rendszert tud indítani, például az OS/2-t vagy a Windowst. Három BIOS-hívás használatával képes megtalálni a számítógépben lévő összes memóriát, és az eredményt jelenti a rendszermagnak. A továbbiakban tehát szükségtelen, hogy a felhasználók a `mem=`*`xxxm`* értéket kézzel írják be, a memória mennyiségének felderítése ugyanis önműködően zajlik. Amennyiben az adott gépen támogatott, a GRUB használja a logikai blokkcímzést (LBA), ezzel sikerült például az „1024 cilinder”-bonyodalmat megoldania, mellyel szemben a LILO tehetetlennek bizonyult.

A felhasználók szempontjából ez azt jelenti, hogy a GRUB az egész merevlemez elérésére képes, azaz bárhonnan tud operációs rendszert indítani. Nem tudom kellően hangsúlyozni, hogy e két utóbbi GRUB-tulajdonság milyen fontos a számomra! S valóban, *Gordon Matzigkeit* GRUB-fejlesztő némi szakmai elfogultsággal két részre osztja a PC-felhasználókat, így LILO- vagy GRUB-rendszerek használóiként jellemzi őket. Szerinte a számtalan program között az indítóprogramok a legfontosabbak, ezért róluk kellene elnevezni az operációs rendszereket. Lehet, hogy szélsőséges a véleménye, de mindenképpen elgondolkodhatunk rajta, hogy miért „Linux” a Linux, hiszen a szó csak a rendszermagot takarja, és nem az egész rendszert. Talán igazságosabbak lennénk a GNU-fejlesztőkkel, ha „GNU/Linux”– vagy „GNU/Hurd”-rendszerekről beszélnénk, együtt említve a rendszermagot és a körülvevő rendszert.

A GRUB telepítése

Én a GRUB-bal a *Linuxvilág* májusi számának Progeny Debian CD-mellékletén találkoztam először, amikor a rendszer telepítése során dönthettem, hogy a GRUB-ot vagy valami mást választok-e rendszerindító programként. Némi tétovázás után a GRUB-ot jelöltem be. Később olvastam el az angol nyelvű leírást, és elsőre mindent meg tudtam csinálni vele. Ezt a nem túl terjedelmes segédletet az `info grub` paranccsal hívhatjuk elő, amit HTML-formában is olvashatunk a /usr/share/doc/grub könyvtárban.

Az írásomban említett összes fájlhoz a megadott útvonal a Progeny Debian faszerkezetében elfoglalt helyekre utal, ugyanezek a fájlok azonban más változatokban esetleg máshol helyezkedhetnek el.

A Progeny Debian-rendszerben a GRUB alapértelmezett, és a `grub_0.5.96.1progeny5.deb` csomagban található. Az újabb változatra vágyók nézzenek körül az ➲ ftp://alpha.gnu.org/gnu/grub FTP-kiszolgálón. A letöltendő fájl neve mindig a következő formátumú: `grub-`*`változat`*`.tar.gz`, tehát az általam használt `grub-0.5.96.1.tar.gz` helyett nagyobb változatszámot kell keresned! Azt is látjuk, hogy a GRUB alfaszakaszban formálódó program, még messze nem éri el az 1.0-s változatot. Nyilván ez a magyarázata annak, hogy csak most kezd felbukkanni a különböző terjesztésekben.

A `tar` programmal becsomagolt, majd a `gzip` által tömörített `tar.gz` kiterjesztésű csomagokat többféleképpen is kibonthatjuk. A művelet két lépésben tehető meg:

```
gunzip grub-0.5.96.1.tar.gz
```

Ez kicsomagolja a tömörített fájlt, és egyetlen tar kiterjesztésű fájlt hagy maga után, amit második lépésben a `tar` programmal kell feldarabolnunk:

```
tar -xvf grub-0.5.96.1.tar
```

A `gunzip` parancs egyébként egyenértékű a `gzip -d grub-0.5.96.1.tar.gz` alakkal. A `gzip` a tömörítő program, amit azonban a `-d` vagy a `--decompress` vagy az `--uncompress` kapcsolókkal kicsomagolásra is használhatunk. A fenti két lépést össze is vonhatjuk, mivel a tar program alkalmas rá:

```
tar -xzvf grub-0.5.96.1.tar.gz
```

Itt a `-z` kapcsoló jelzi a `tar` programnak, hogy a `-f` (azaz fájl) kapcsoló által kijelölt fájlt a `gzip`-pel kell kitömörítenie. Használhattuk volna még a `-z` kapcsoló helyett a hosszabb `--gzip` vagy `--ungzip` kapcsolókat is. Például:

```
tar --extract --verbose --ungzip
➥--file grub-0.5.96.1.tar.gz
```

(Az `extract` kapcsoló az `x`, a `verbose` kapcsoló pedig a `v` helyett áll.) Egzotikusabb forma a `zcat`. Ez egyenértékű a `-c` vagy `--stdout` vagy `--to-stdout` kapcsolóval ellátott `gzip` paranccsal. Ezek a kapcsolók arra kényszerítik a `gzip` programot, hogy kimenetét a parancssorra irányítsa:

```
zcat grub-0.5.96.1.tar.gz
```

Ezzel nem sokra megyünk, főként nagy fájlok esetén, ezért célszerű a `zcat` kimenetét a `|` [szűrő (pipe) karakter] használatával egy csővezetékben a tar programhoz átirányítani:

```
zcat grub-0.5.96.1.tar.gz | tar
➥-xv
```

A `-` kapcsoló jelzi a `tar`-nak, hogy ne keressen fájlt. Egy másik változat:

```
gzip -d < grub-0.5.96.1.tar.gz
➥| tar -xp
```

A tar `-p` kapcsolója megőrzi a hozzáférési jogokat. A `-p` helyett a `--same-permissions` vagy `--preserve-permission` szavakat is írhattuk volna.

A kibontás után létrejön a grub-0.5.96.1 nevű könyvtár, amely a forrásfájlokat tartalmazza. Át kell lépnünk ebbe a könyvtárba, és itt kell kiadnunk a szokásos fordítási utasításokat:

```
$ cd grub-0.5.96.1
$ ./configure
$ make
$ su
Password:
# make install
```

Ha ez elsőre nem történik meg, akkor még az INSTALL fájlt is elolvashatjuk, de ha nagyon nem megy, célszerűbb lesz DEB- vagy RPM-csomagokat keresnünk a Világhálón. Ezek az utasítások a megfelelő helyekre rakják szét a fájlokat, de a GRUB-ot telepíteni kell. Unix-típusú rendszerekben írtak ehhez egy segédeszközt is , a `/usr/sbin/grub-install` programot. Használata igen egyszerű:

```
# grub-install /dev/hda
```

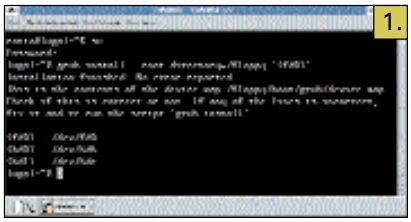
Ez az első IDE-merevlemez rendszerindító területébe (Master Boot Record – MBR) fogja tenni a GRUB-ot. A sor elején található `#` karakter mutatja, hogy mindezt rendszergazdai jogosultsággal tehetjük meg. Nem feltétlenül szükségszerű, de szokásos, hogy a Bash-héjban a rendszergazdai (root) jogosultságot a parancssoron a `#` karakter jelzi, míg a felhasználók a `$` készenléti jelet kapják. Fentebb láthattuk, hogy a beállítást és a fordítást egyszerű felhasználóként végeztem, de a `make install` parancsot már rendszergazdai jogosultsággal kellett kiadnom, hiszen a rendszer területére írtam. Amennyiben csak a saját HOME könyvtáramba telepítettem volna, természetesen nem lett volna szükségem a rendszergazdai jogokra. Ha eredetileg nem rendszergazdaként jelentkezünk be, akkor ezt a jogot később is elnyerhetjük a `su` parancs kiadásával, miután a következő sorban beírjuk a rendszergazdai jelszót.
Ha telepítéskor külön indító lemezterületet hoztunk létre, ennek helyét meg kell mondanunk a GRUB-nak, máskülönben nem fogja megtalálni az indítókönyvtárat:

```
# grub-install --root-
➥directory=/boot /dev/hda
```

Ugyanez a helyzet, ha hajlékonylemezen hozunk létre fájlrendszert:

```
# superformat /dev/fd0 hd
# mkfs -t minix /dev/fd0 1440
# mount /dev/fd0 /floppy
# grub-install --root-
➥directory=/floppy '(fd0)'
# umount /floppy
```

A fenti utasítássor először megformázza az üres hajlékonylemezt, majd minix fájlrendszert készít rajta. (Használjuk a `# mke2fs ➥/dev/fd0` parancsot, ha ext2 fájlrendszert szeretnénk!) A következő sor `mount` parancsa csatolja a hajlékonylemezt a korábban, a rendszer telepítésekor alapértelmezetten létrehozott /floppy könyvtárba, majd a `grub-install` parancs átmásolja a /floppy/boot/grub könyvtárba a szükséges fájlokat, melyek alapértelmezetten a /usr/share/grub/i386-pc könyvtárban találhatók. Ezeket a fájlokat a grub-install az ekkor létrehozott `device-map` fájllal egészíti ki. Az utolsó sorban az `umount` paranccsal lecsatoljuk a hajlékonylemez-meghajtót.


1.

Az *1. képen* látható `grub-install` parancs még nem hoz létre valódi indítólemezt. A GRUB leírása szerint ezt a következőképpen kell megtenni:

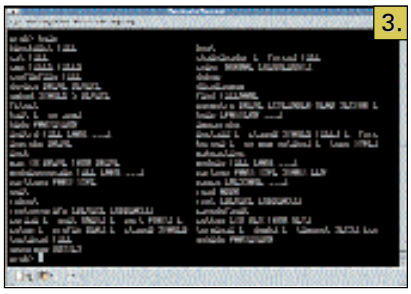
```
# cd /boot/grub
# dd if=stage1 of=/dev/fd0
➥bs=512 count=1
1+0 records in
1+0 records out
# dd if=stage2 of=/dev/fd0
➥bs=512 seek=1
153+1 records in
153+1 records out
#
```


2.

A `dd` parancs a bemeneti állományt (`if=xxx`) a kimeneti állományba (`of=xxx`) másolja át, de a `cp` másolóparanccsal ellentétben a `dd` esetében megadhatjuk a másolandó blokkméretet (`bs=xxx`), illetve másolás közben további változtatásokat is végrehajthatunk. Rendszergazdai jogon lefuttattam a fenti parancsokat, de a második lépésben a *2. képen* látható hibaüzenetet kaptam: Végül a fenti hiba miatt inkább a GRUB parancssort használtam.

A parancssor

Ha rendszergazdai jogosultsággal meghívjuk a `/usr/sbin/grub` programot, a GRUB parancssorba jutunk. Ennek formai követelménye hasonló a Bash parancssorhoz, de annyira azért nem kimunkált, mint nagynevű elődjéé. Ismerkedés céljából elsőként írjuk be a `help` szót. A *3. képen* láthatjuk, hogy jó néhány ismerősnek tűnő parancs közt pár


3.

egzotikusabb is kiírodik. A parancsok rövid leírása a /usr/share/doc/grub/grub_13.html fájlban olvasható. Érdemes rögtön megjegyeznünk a `quit` parancsot, mert ezzel léphetünk ki a GRUB-héjból. A `reboot` például újraindítja a rendszert, a `root` paranccsal pedig azt az eszközt és lemezterületet adhatjuk meg, ahová a gyökérkönyvtárat feltelepítettük. Ha például a gyökérkönyvtár a hajlékonylemezen található, írjuk ezt:

```
grub> root (fd0)
 Filesystem type is ext2fs,
  using whole disk
```

Ha a második lemez harmadik lemezterületén:

```
grub> root (hd1,2)
 Filesystem type is reiserfs,
 partition type 0x83
```

Az `'(fd0)'` kifejezés egyenértékű a Linux-rendszerben használatos /dev/fd0 írásmóddal; olyan, mintha a korábbi példában nem `grub-install /dev/hda`-t írtunk volna, hanem `grub-install '(hd0)'`-t.
Ezek a leképezett értékek kiolvashatók a /boot/grub/device.map fájlból, és inkább a meghajtók BIOS-beli helyzetére, mint a rendszerben alkalmazott számozásra vonatkoznak. A device.map fájl nálam a következőképpen néz ki:

```
(fd0)   /dev/fd0
(hd0)   /dev/hdb
(hd1)   /dev/hde
```

Az első oszlop tartalmazza a BIOS-eszközöket. Ezek az általam adott példában az A, C és Ext meghajtóknak felelnek meg, azaz a hajlékonylemeznek, az IDE- és az Ultra ATA-meghajtónak. A vizsgált gépen ezek mindegyikéről történhet rendszerindítás. A második oszlop a fájlneveket foglalja magába. Tudjuk, hogy a Unixban minden fájl, így a rendszer, a fenti eszközöket különleges állományokként kezeli. Azért szükséges ez a hozzárendelés, mert a GRUB nem mindig ismeri fel megfelelően az eszközöket, és ilyenkor szerkesztenünk kell a device.map fájlt. Látjuk, hogy a GRUB formai követelményei szerint az eszközök neveit `( … )` zárójelek közé kell tennünk. A hajlékonylemez azonosítója esetében az fd magát a meghajtót jelenti, a mögötte lévő szám pedig azt, hogy hányadik meghajtó. A hd merevlemezt jelent, tekintet nélkül arra, hogy IDE-, Ultra DMA- vagy SCSI-lemezről van-e szó. A mögötte álló szám pedig azt jelzi, hogy az első, második vagy harmadik stb. meghajtóval van-e dolgunk. A számozás nullával kezdődik, tehát az első meghajtó száma 0, a másodiké 1 és így tovább. A merevlemezeken található lemezterületeket (partition) újabb számmal különböztetjük meg. Az első lemezen lévő második lemezterületet például így jelöljük: `(hd0,1)`. A lemezterületek számozása ugyancsak nullával kezdődik. Az első kibővített (extended) lemezterületen

létrehozott logikai meghajtó száma 4, azaz a következőképpen jelöljük: `(hd1,4)`.
A GRUB a fájlokat is e jelölés segítségével találja meg. Teszem azt, ha a GRUB az első merevlemez harmadik lemezterületére lett telepítve, akkor az ott található device.map fájlra így hivatkozhatunk:

```
(hd0,2)/boot/grub/device.map
```

Ez az abszolút címzési mód. Ha a `root` paranccsal (lásd fentebb) korábban már tudtára adtuk a GRUB-héjnak, hogy a / gyökérkönyvtár melyik eszközön helyezkedik el, akkor használhatunk relatív címzést is, mivel a GRUB számára már ismert a példában szereplő `(hd0,2)` eszköz. Ilyenkor elég ennyit írni:

```
/boot/grub/device.map
```

A Unix-változatokban elvárható, hogy egy magára valamennyire is adó héjban Ctrl+I vagy a Tab billentyű megnyomásakor parancskiegészítést kapjunk. Nincs ez másképp a GRUB-héjban sem. Ha például csak annyit gépelünk, hogy `ker`, majd leütjük a tabulátort, akkor a parancs `kernel`-re egészül ki. Ha több lehetséges parancs közül választhatunk, természetesen listát kapunk:

```
grub> h
 Possible commands are: halt
 help hide
```

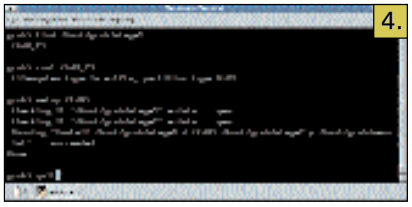
Ha mondjuk nem tudjuk, milyen meghajtónevek találhatók a gépben, a `find (` parancs begépelése után megnyomhatjuk a Tab billentyűt, hogy megjelentessük a lehetséges lemezek listáját:

```
grub> find (
 Possible disks are:  fd0 hd0
 hd1
```

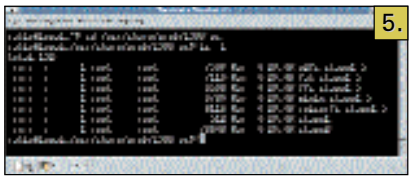
GRUB-parancsok futtatásakor is követhetünk el hibákat, ilyenkor ehhez hasonló hibaüzeneteket kapunk:

```
grub> root (fd0,)

Error 12: Invalid device
 requested
```


4.

A lehetséges hibaüzenetetek listája a /usr/share/doc/grub/grub_14.html fájlban található.
Miután a `dd` paranccsal nem tudtam indítólemezt készíteni, a parancssoron próbáltam meg a kézikönyvben leírtak szerint. Először a GRUB-héj `find` parancsával megnéztem, hogy a stage1 fájl hol található. A kapott `(hd0,2)` eredmény mutatja, hogy az első merevlemez harmadik lemezterületén volt. A másodszor kiadott `root` parancs tudatja a GRUB-bal, hol helyezkedik el a gyökérkönyvtár. Ezt követően már lefuttathattam a `setup` parancsot, amely az `install` parancs segítségével telepíti a GRUB-ot a kapcsolójában megadott `(fd0)` lemezre. Ha a `setup (hd0)` parancsot adtam volna ki, akkor a GRUB az MBR-be került volna. Ha az első lemezrész indítószektorába szerettem volna tenni, a parancs a következőképpen nézett volna ki: `setup (hd0,0)`. Amennyiben a GRUB-ot nem az első meghajtó első lemezterületére tesszük, azt több lépcsőben (`chainload`) kell egy másik rendszerindító programmal betölteni. Azért beszélünk ilyenkor chainloaderről (azaz láncolt betöltőről), mert az egyik rendszerindító első láncszemként tölti be az utána következő második láncszemet, a másik operációs rendszer indítóprogramját, esetleg egy másik köztes indítót.
A GRUB telepítésekor először az 512 bájt nagyságú első szakasz, azaz a stage1 fájl másolódik be az MBR-be vagy az indítóterületre a /usr/share/grub/i386-pc könyvtárból. Azért pontosan 512 bájt a mérete, mert ennyit és nem többet lehet ezekre a területekre átmásolni. Ennek a programnak mind-


5.

össze az a feladata, hogy a másfeledik (stage1_5) vagy a második (stage2) szakaszt betöltse. A másfeledik szakaszból több is akad, és mindegyikük felismeri a rá jellemző fájlrendszert, amit a nevükből is rögtön ki lehet találni.
Ezeknek a fájloknak szintén viszonylag kicsi a méretük, ezért rögzített helyre írhatók, közvetlenül a stage1 fájl után. Az adott fájlrendszerre jellemző stage1_5 programszakasz fogja betölteni az immár terjedelmesebb, 78848 bájt méretű stage2 fájlt, amely a GRUB magját tartalmazza.

A menüfelület

Rendszerindításkor arra is lehetőség nyílik, hogy rövid menü jelenjen meg, amely megmutatja a felhasználóknak, hogy a le és fel nyilak segítségével milyen operációs rendszerek közül választhatnak. A kiválasztott sor kivilágosodik, és az Enter leütésekor megkezdődik a rendszerindítás. Ez a lehetőség a LILO esetében is adott. Ami ehhez képest új a GRUB-ban, az az, hogy ezt a menüt menet közben is lehet szerkeszteni, azaz még az indítás szakaszában. Előnyös tulajdonság, hiszen ha a lilo.conf fájlban elírtam valamit, annak komoly büntetése volt, mivel csak a rendszer teljes újraindítása után nyílt lehetőségem a javításra. A GRUB menüben a szerkeszteni kívánt sor a le és fel nyilakkal választható ki, amit az `e` betű leütése követ (feltételezem, az „e” az edit („szerkeszt”) szó első betűjéből ered).
A megjelenő sorokból szintén választani kell, majd ismét az `e` következik, ha a kiválasztott sort szeretnénk szerkeszteni. Ha a szerkesztett sort jónak találjuk, használjuk az Enter, ha nem, az Esc billentyűt. A GRUB leírása szerint az Esc megnyomása egyben visszavonásként (`redo`) is működik, mivel a menüben rögtön az előző állapothoz kerülünk vissza. A kis `o` betű leütésével az éppen szerkesztett sor mögé szúrhatunk be egy sort, a nagy `O`-val pedig a sor elé. Ha a `d` betűt használjuk, az egész sor törlődik. Természetesen egyszerűbb a menüt egy szövegszerkesztőben megírni és javítani. A menü megváltoztatásához a /boot/grub/menu.lst fájlt kell átírni. A vizsgált gépen ez a fájl a következőképpen néz ki:

```
timeout 10
default 0

# --> PROGENY START (1.0) <--

title Progeny Debian (kernel
➥2.4.2)
root (hd0,2)
kernel /boot/vmlinuz-2.4.2
➥root=/dev/hdb3 ro
initrd /boot/initrd-2.4.2.gz

title SuSE 7.0 hdb2 (kernel
➥2.2.16)
root (hd0,1)
kernel /boot/vmlinuz
➥root=/dev/hdb2 ro
initrd /boot/initrd

title Windows Millenium
root (hd0,0)
makeactive
chainloader +1

title Progeny Debian (kernel
➥2.2.18)
root (hd0,2)
kernel /boot/vmlinuz-2.2.18
➥root=/dev/hdb3 ro
initrd /boot/initrd-2.2.18.gz

title Progeny Debian - single-
➥user (kernel 2.4.2)
root (hd0,2)
kernel /boot/vmlinuz-2.4.2
➥root=/dev/hdb3 ro single
initrd /boot/initrd-2.4.2.gz

# --> PROGENY END <--
```

Az első sor timeout 10 kifejezése mondja meg a rendszerindító GRUB-nak, hogy a menüt 10 másodpercig jelenítse meg, utána önműködően indítsa el a `default` parancs értékében meghatározott menüpontot, hacsak a felhasználó másként nem rendelkezik. A 0 érték az első menüpontot jelenti, így példánkban ez az alapértelmezett, de más számot is választhattunk volna. A # jel mögött találhatók a megjegyzések. A `title` parancs mögötti szöveg lesz olvasható a menüben. A `root (hd0,2)` paranccsal megadjuk, hogy melyik meghajtó melyik lemezterületén található a gyökérkönyvtár, a `kernel` parancs betölti a kívánt rendszermagot, az `initrd` pedig a meghatározott initrd fájlt használja indításkor. A `kernel` parancs második részében a rendszermag parancssori (kernel command line) ismert értékeit adhatjuk meg, amelyeket a LILO-nál az `append` fűzött a többihez.

A Windows indítása ettől annyiban tér el, hogy a GRUB ezt az operációs rendszert nem közvetlenül indítja. Először a `root` vagy a `rootnoverify` paranccsal meghatározza a Windows helyét, majd a `makeactive` paranccsal kijelöltí azt a lemezterületet, és végül elindítja az ottani rendszerindító programot. Itt feltételezzük, hogy az indítóprogram (boot loader) ugyanannak a lemezterületnek az indítóterületén (boot sector) található, ahová magát az operációs rendszert is telepítettük. Mivel a Windows a láncolt indítás szempontjából nem támogatott operációs rendszer, esetében használhatjuk a `rootnoverify (hd0,0)` parancsot, amely a `root` paranccsal ellentétben nem próbálja meg befűzni a lemezterületet azért, hogy meghatározza annak méretét és a meghajtó típusát.

A /usr/share/doc/grub/menu.lst egy mintafájl, amely jól használható példa annak megmutatására, miképpen kell a GNU/Hurd, a Linux, a Mach, a FreeBSD, az OS/2, a DOS és a Windows operációs rendszereket indítani. Az ugyanitt lévő grub_5.html fájlban olvashatunk róla egy nem túl bőbeszédű ismertetőt.

Játékosabb kedvű GRUB-felhasználók a `color` parancs segítségével megváltoztathatják a menü színeit a menu.lst fájlban, vagy a GRUB parancssoron:

```
title Villogok
color yellow/green blink-light-
➥cyan/magenta
```

A színek párban állnak. Az elöl lévő szín az előtér, a hátulsó a háttér. A `color` parancs utáni első színpár a normál szín, a második pedig a kiemelt szín. Az előtér és a háttér színei a következők lehetnek: black (fekete), blue (söétkék), green (zöld), cyan (égszínkék), red (vörös), magenta (bíbor), brown (barna), light-gray (világosszürke). A most következő színek viszont csak az előtér színeiként használhatók: dark-gray (sötétszürke), light-blue (világoskék), light-green (világoszöld), light-cyan (kék), light-red (világos piros), light-magenta (világos bíborvörös), yellow (yellow), white (fehér). Ha a szín elé a blink szót írjuk, az előtérszín villogni fog.

A menu.lst fájlba beírt változtatások érvényesítéséhez semmilyen parancsot nem kell lefuttatni, azok a következő indításkor életbe fognak lépni.

A begépelt parancsok és kapcsolók a következő billentyűkombinációkkal szerkeszthetők

A Ctrl-P vagy a felfelé nyíl visszahozza a korábban beírt parancsokat.
A Ctrl-F vagy a jobbra nyíl egy karakterrel előremozgat.
A Ctrl-B vagy a balra nyíl egy karaktert hátraugrik.
A Ctrl-A vagy a Home a sor elejére ugrik.
A Ctrl-E vagy az End a sor végére mozgat.
A Ctrl-D vagy Delete törli a kurzor alatti karaktert.
A Ctrl-H vagy backspace a kurzortól balra lévő karaktert törli.
A Ctrl-K törli a szöveget a kurzor jelenlegi helyzetétől a sor végéig.
A Ctrl-U törli a szöveget a kurzortól a sor elejéig.
A Ctrl-Y beszúrja a törölt szöveget a kurzor pozíciójába.

Kipróbálás

A `grub` héjparancsainak igazi haszna nem a rendszer futása során, hanem rendszerindításkor válik nyilvánvalóvá. Ha a `grub`-ot telepítettük, annak jó ismerői a rendszer indulásakor sokfélét megtudhatnak a gépről, és a kipróbálás céljából különböző parancsokat hajthatunk végre. A `find` paranccsal például kiírattathatják az összes meghajtót:

```
grub>  find (
       Possible disks are:
        fd0   fd1   hd0   hd1
```

Fájlokat listázhatunk ki:

```
grub>  cat
    ➥(hd0,2)/boot/grub/menu.lst
```

vagy tetszőleges kapcsolókkal indíthatjuk a rendszert, mint például az alábbi esetben `grub` parancssorból indítottam a rendszert:

```
grub> kernel
➥(hdb0,2)/boot/vmlinuz-2.4.2
➥root=/dev/hdb3 ro mem=128M
grub> initrd
➥(hdb0,2)/boot/initrd-2.4.2.gz
grub> boot
```

Bizonyos esetekben a rendszermag betöltése után szükséges lehet a `module` vagy `modulenounzip` parancsok kiadása is, hogy modulokat töltsünk be. Az utóbbi nem csomagolja ki a modulokat önműködően. Figyeljük meg, hogy a `mem=128M` rendszermag parancssori értékkel meghatároztam a rendszer számára látható memória mennyiségét, a valós értéktől függetlenül. Ez olyankor lehet hasznos, amikor a programírás során arra vagyunk kíváncsiak, miként viselkedik egy nagy teljesítményű gépen fejlesztett, memóriaigényes program, ha negyedannyi memória jut neki.

Segédletek

A /usr/share/grub könyvtárban található még egy *Jeff Licquia* által írt lilo2grub nevű Python parancsfájl, ami a LILO lilo.conf beállítófájlt fájlt a GRUB számára is fogyasztható menu.lst alakra hozza.

A nuni

Nem állíthatom, hogy sikerült mindent elmondanom a GRUB rendszerindító programról. Például nem érintettem a biztonsági kérdéseket vagy a hálózati indítási lehetőségeket. Azt sem tanácsolom senkinek, hogy eddig jól működő LILO-ját most törölje, és helyébe a GRUB-ot tegye, hiszen ennek nem sok értelme lenne. Ugyancsak nem állítom, hogy csak a LILO és a GRUB versenyez a rendszerindító programok között. Ott van például a nuni, ami egy kicsi, Linuxra írt rendszerindító az ext2 fájlrendszerre és IDE-meghajtókra. A nuni is túl akar lépni az 1024 cilinderes határon, és 128 GB-ig bárhonnan indíthat a meghajtóról. Kissé terjedelmesre sikerült, ezért pillanatnyilag csak hajlékonylemezekre telepíthető. Fejlesztője az amerikai Oregonban élő *Neil Koozer* (nkoozer@rosenet.net).

Szaló István (ratiosoft@freemail.hu) tanár, immár több mint másfél évtizede foglalkozik programozással, de csak a Java és a Linux megismerése után tudta meg, hogy mi is az igazi programozás. Azóta hátat fordított a feketedoboz módszereknek, és most szabadidejében a nyílt forráskódú Java és a GNU C vagy C++ programokat tanulmányozza. Több írása megjelent már a hazai számítástechnikai lapokban. Ha néha feláll számítógépe mellől, rendszerint művészettörténész feleségével és kisiskolás lányával „találja szemben" magát.