A Linux testreszabása az alapoktól

Ha nem találunk kedvünkre való Linux-változatot, készítsük el sajátunkat!

Sokféle Linux-változattal találkozhattunk már az eddigiekben, és talán még több lesz. Némelyik apró, mint a DLX a maga egyetlen hajlékonylemeznyi méretével, némelyek pedig kimondott óriások, mint a Red Hat 6.2, mely öt CD-t is megtölt. Úgy tűnik, minden egyre bonyolultabbá válik, ahogy a rendszerek mérete nő. Egyre kevésbé érthető, hogyan képes a rendszer innen-onnan származó kódok darabjaiból összeállni, és egyre kisebb reményt látunk arra, hogy képesek legyünk saját rendszerünk összeállítására és testreszabására bizonyos célok szem előtt tartásával.
A dolog szerencsére nem ilyen kilátástalan, hiszen az alaprendszer szemszögéből nézve minden változat hasonlóan áll össze. Az egyetlen különbség, hogy a nagyok több programcsomaggal és kiegészítővel szélesebb közönség megnyerésére törekednek, míg a kicsik kevesebb eszközükkel szűkebb felhasználói kört céloznak meg.
A széles körű képességekkel bíró Linux-változatok többnyire szükségtelenül nagyok az egyedi alkalmazások igényeihez. Vegyük például a beágyazott rendszereket, ezek nem támasztanak túl nagy igényt egy-egy feladat megvalósításán túl a Linux-rendszerrel szemben. Mindemellett a lehetőségek és az igények olyannyira különbözőek lehetnek, hogy a gyártók képtelenek olyan változatot készíteni, mely kellően átfogó, egyszersmind a legtöbb felhasználó elvárásainak megfelelő. Az alkalmazások a hatékony működéshez rendszerint testre szabott alaprendszert kívánnak. Ez egyszerűen megoldható, csak fizetnünk kell érte valamely változat gyártójának. Van azonban más megoldás is – végezzük el a testreszabást magunk! Ez a „csináld magad” módszer nem csupán nagyszerű mulatság, de sok esetben létfontosságú megoldást nyújt, így egyre többen értékelik a Linux megbecsülendő lehetőségeként. Ha itt tapasztalatokat nyerünk, nemcsak a rendszermag testreszabását tanulhatjuk meg, hanem rendszerünk többi összetevőjét is képesek leszünk beállítani az igényeinknek leginkább megfelelő teljesítmény eléréséhez.
Írásomban szeretném megmutatni, hogy saját Linux-alaprendszerünk elkészítése nem lehetetlen feladat. Megosztom e téren szerzett tapasztalataimat. Alaprendszerről van szó, mely tehát kicsiny, átlátható és működőképes. Megkíséreljük egyszerűvé tenni mindazt, ami bonyolultnak látszik. Lépésenként haladunk a felépítésben, a testreszabással pedig elérjük, hogy a rendszer akár egy hajlékonylemezen is elférjen. Ha mindennek a végére jutottunk, a kapott összeállítás jó kiindulási pont lehet az egyes alkalmazások igényeinek megfelelő alaprendszer kiépítésében.

A Linux-rendszer

A rendszerek általában több, egymással kapcsolatban álló részből tevődnek össze. A Linux esetében ezek két fő területre csoportosíthatók: a rendszermag, valamint a többi összetevő, amelyek nélkül a rendszermag semmire sem volna jó. A rendszermagon kívül minden programelem fájlrendszerben kap helyet. Így a gyakorlat szempontjából a Linux-rendszert felfoghatjuk úgy, mint a rendszermag és a fájlrendszer egységét – és ez alól egyik Linux-változat sem kivétel. Így például egy teljes egészében telepített Linux-változat egyszerűen egy rendszermagból és egy hatalmas fájlrendszerből áll. A Linux telepítőlemezeivel a teljes rendszert telepíthetjük, míg egy vészrendszer az esetleges hibák bekövetkeztekor lehet életmentő segítség. Mindkettő hasonló módon épül fel, vagyis rendszermagot és egy kezdeti fájlrendszert tartalmaznak – ez utóbbi azonban kicsiny, így csak néhány feladat elvégzéséhez szükséges eszközöket tartalmaz (lásd az *1. ábrát*). A Linux rendszermag alkalmas a fájlrendszer felkutatására és üzembe helyezésére, legyen az hagyományosan egy lemezrészre telepítve vagy egy tömörített lemezlenyomatba összepréselve.

1. ábra A Linux-rendszer

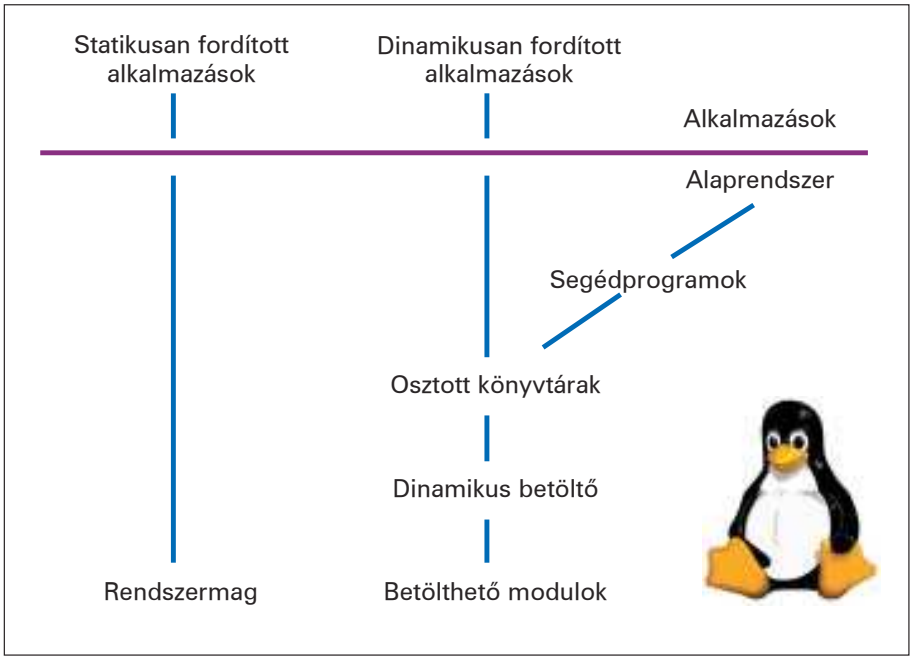
Mi az alaprendszer?

Tegyük fel, hogy van egy alkalmazásunk, melyet egy osztott könyvtárakat használó rendszermaggal szeretnénk futtatni. Tételezzük fel továbbá, hogy ez az alkalmazás nem igényel semmiféle különleges lehetőséget, mellyel a rendszer jelenleg nem rendelkezik. Egy olyan alaprendszert szeretnénk létrehozni, amelyen futtathatjuk ezt az alkalmazást, emellett működését némiképpen szabályozhatjuk is.
A *2. ábra* egy jellegzetes alaprendszert ábrázol – meg kell azonban jegyeznünk, hogy a kép nem teljes, hiszen egy program lehet statikusan fordított is, melynek így nincs szüksége az osztott könyvtárakra, továbbá lehet a.out típusú is, ami viszont azt jelenti, hogy nincs szüksége a dinamikus betöltőre.
Ha az alkalmazás „önálló”, vagyis minden futás közben szükséges segédeszközzel statikusan lett fordítva, akkor nincs szüksége az osztott könyvtárakra, így közvetlenül a rendszermag fölött futhat – az alaprendszer tehát ilyen esetben magát a rendszermagot jelenti. Mindazonáltal szinte minden rendszerben szükség van bizonyos segédeszközök támogatására, legyen szó akár a fájlműveletekről, akár a rendszermegfigyelésről (gondolhatunk itt a mount, illetve a ps parancsokra). Alaprendszerünk felépítésébe tehát beleértjük a rendszermagot, a dinamikus betöltőt, valamint néhány elengedhetetlenül szükséges könyvtárat és segédeszközt.
Célunk egy olyan alaprendszer kiépítése, mely elfér egy hajlékonylemezen, és tartalmazza a rendszermagot, valamint egy tömörített kezdeti fájlrendszert (lásd a *3. ábrát*). Ezt a tömörített fájlrendszert a rendszermag kicsomagolja, majd elhelyezi a memóriában. Az ilyen és hasonló alaprendszerek készítése gyakran nem nehéz feladat, azonban sokak számára túlságosan hosszadalmas. Nos, ezen fogunk változtatni a következőkben.
A fent vázolt alaprendszer testreszabása nagymértékben függ attól az alkalmazástól, amelynek rendszerünket alá szeretnénk rendelni. Válasszuk ki a rendszermag megfelelő beállításait, egy dinamikus betöltőt, az alkalmazás és a segédprogramok működéséhez szükséges könyvtárakat, valamint olyan alapvető segédeszközöket, melyek a rendszer szabályozásához és fenntartásához szükségesek. Ezután fordítsuk le mindezeket egy megfelelő környezetben, csomagoljuk be az eredményt, majd tegyük az egészet indíthatóvá. Ha valami még hiányzik, kezdjük újra az egészet, belevéve az előzőekben kimaradt összetevőt is.

Szerény célkitűzéseink

A kíváncsi olvasó megkérdezheti, miért tesszük mindezt, és vajon mire jó ez az egész? Nos, másokhoz hasonlóan mi is szeretnénk összetett alkalmazásokat futtatni, mondjuk többszálas többfeladatos programokat egy jellemzően PC-szerű gépen, amelyben azonban nincsen merevlemez, vagy esetleg nem kapcsolódik hozzá képernyő. Szükségünk van tehát egy rendszermagra, és néhány egyéb elemre, amelyekkel elkezdhetünk kísérletezni. Az így kapott rendszertől három fontos dolgot követelünk meg: legyen megbízható, fenntartható és testre szabható. Sokunk számára egy saját rendszermag létrehozása e célra némiképp túlméretezett feladatnak tűnhet. Itt lép be a képbe a Linux, és a nyílt forrás társadalma – együtt megkímélnek a terhes feladattól.

Alapanyagaink tehát készen állnak, ráadásul ingyenesen beszerezhetők. Feladatunk csak annyi, hogy kiválasszuk a megfelelő építőkockákat és felépítsük belőlük saját rendszerünket.



2. ábra Az alaprendszer összetevői

Mielőtt azonban mindennek nekikezdenénk, néhány alapvető kérdésre választ kell kapnunk: Hogyan fordítsuk le a rendszermagot? Vagy egy osztott könyvtárat? Miként készíthetjük el a kezdeti fájlrendszert? Hogyan helyezzük el a rendszermagot és a tömörített fájlrendszert egy hajlékonylemezen vagy EPROM-ban? Miként futtathatjuk az osztott könyvtárakat használó alkalmazásokat? Hogyan végezzük a hibakeresést? Számos ilyen és hasonló kérdést tehetnénk még fel. A válaszok pedig, ha nem is egy helyen, de fellelhetők. Ami most következik, az saját tapasztalataimon alapul, és a sorok között, ha rejtve is, választ találunk sok fenti kérdésre.

A kivitelezés lépései

Ha a testreszabás tervei előttünk állnak, megkezdhetjük a kivitelezés lépéseit. Kezdjük az általános tennivalókkal:

1. A fejlesztőkörnyezet üzembe helyezése.

Fejlesztőkörnyezetként telepítsünk egy teljes Linux-változatot, mint például a Red Hat 6.0. Győződjünk meg arról, hogy a gcc támogatja az adott rendszert. A dolgok egyszerűsítése végett tegyük fel, hogy célgépünk (melyen majd a testre szabott rendszer működik), valamint gazdagépünk (melyen a fejlesztést végezzük) ugyanolyan típusú processzorral működik, vagyis esetünkben Intel x86-tal, egyébként elő kell készítenünk egy keresztfordítót.

2. A rendszermag beállítása.

Szerezzük be a legfrissebb megbízható (stable) rendszermag forrását (a cikk írása idején ez a 2.2.17-as változat). A beállításokról és a fordításról a forrásfájlok megfelelően tájékoztatnak, így az ott leírtakat itt nem ismételjük meg. Azt azonban meg kell jegyeznünk, hogy szükség lesz a kezdeti memórialemez, valamint a betölthető modulok támogatására és még számos beállításra. Ha alaprendszerünkben valamilyen egyedi eszközt – például egy hálózati kártyát – kívánunk használni, modulként támogathatjuk. Ezeket a modulokat azután szabadon telepíthetjük és használhatjuk az alaprendszerben.

3. Az alapkönyvtárak előkészítése.

Szerezzük be a glibc legfrissebb változatát (jelenleg ez a glibc2 2.1.3). Itt szinte mindent megtalálhatunk, amire szükségünk lesz – a dinamikus betöltőt, a szabványos C könyvtárat, a matematikai könyvtárat és másegyebeket. Mindenre természetesen nem lesz szükségünk, azonban jobb egyszerre előkészíteni az egészet, és azután kiválasztani a megfelelőket. A glibc források között található leírások elegendő útmutatást adnak a fordításhoz. Mivel azonban a könyvtárakat a célgépen fogjuk használni, a fordítást a 2. lépésnek megfelelő rendszermag-fejlécfájlokkal kell elvégeznünk, ehhez a with-headers kapcsolót kell használnunk. Mindemellett a könyvtár fejlécfájljait is telepítenünk kell, hogy a célgép számára fordítandó többi programhoz is elérhetők legyenek.

4. A keresztfordítás beállítása.

A rendszermag és a glibc fejlécfájljainak telepítése után fel kell készítenünk a gcc fordítóprogramot a használatukra. Erről részletesen a Glibc2-HOWTO fájlban olvashatunk. Röviden összefoglalva, teendőnk lényegében az, hogy a -b kapcsolóval megadjuk a gcc-nek, merre találja a meghatározásokat. Esetünkben – mivel a célgép és a gazdagép alapjában ugyanaz – elegendő a gazdagép adatait használnunk, amiket a gcc -v paranccsal deríthetünk ki. Saját géptől esetén ezt a választ kaptam a parancsra:

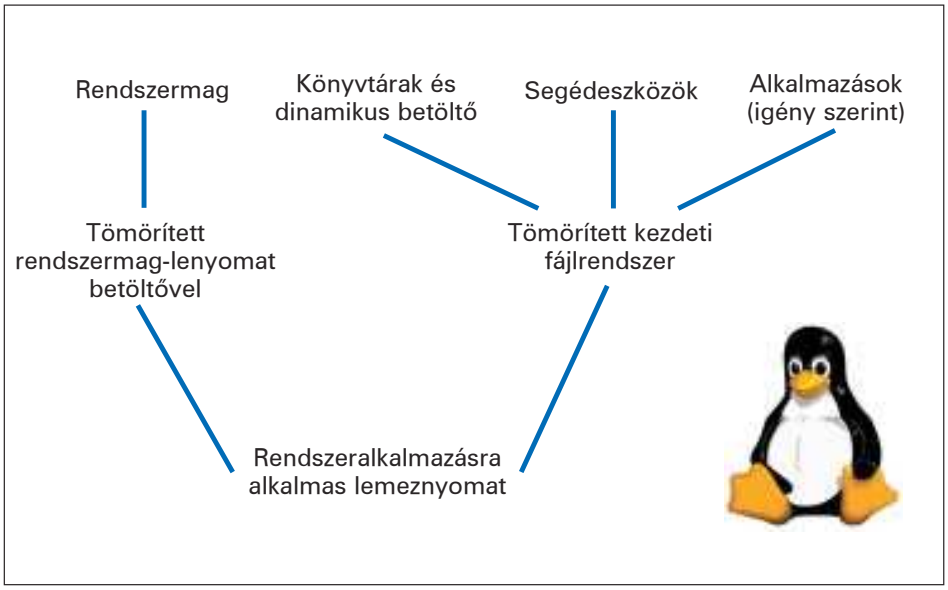
```
Reading specs from
/usr/lib/gcc-lib/i386-redhat-linux/egcs-2.91.66/
specs
gcc version egcs-2.91.66 19990314/Linux
(egcs-1.1.2 release)
```

Fordítsuk le a rendszer többi elemét is a -b kapcsolóval, valahogy így:

```
gcc -b i386-redhat-linux
```

5. A kívánt könyvtárak és segédeszközök kiválasztása és előkészítése.

Egy segédprogram – például a mount – vagy valamely glibc-ben nem szereplő könyvtár, mondjuk a termcap lefordításához mindenekelőtt tudatnunk kell a fordítóval, merre találja a fejlécfájlokat (include) és a szükséges könyvtárakat. Először is, a --nostdinc kapcsoló használatával közöljük a fordítóval, hogy ne keresse a fejlécfájlokat az alapértelmezett útvonalak mentén. Ezután a -b $MACHINE kapcsolóval tudassuk vele azt is, hogy a fordítás a célgép számára készül. Ha ezzel megvagyunk, adjuk meg a rendszermag és a szabványos könyvtár fej-



3. ábra A rendszer elemeinek elrendezése

lécfájlainak helyét a -I kapcsolóval. Végezetül a -L és -l segítségével határozzuk meg, mely könyvtárakat használja a betöltő, és ezeket hol találhatja meg.

6. Az alkalmazások elkészítése.

A könyvtárakhoz és segédprogramokhoz hasonlóan az alkalmazások keresztfordítására is sort kell kerítenünk, ha a célgépen szeretnénk használni őket. A rendszer szempontjából semmiféle különleges intézkedésre nincs szükség, feltéve természetesen, hogy az alkalmazás működéséhez szükséges egyéb elemeket telepítettük.

7. Becsomagolás.

Ha az elemekkel elkészültünk, már csak megfelelő elrendezésükről kell gondoskodnunk, hogy rendszerindításkor minden a helyére kerüljön. A következő szakaszban kiderül, hogyan.

8. További kiegészítések.

Az alaprendszert kiindulási pontként használhatjuk kézzel fogható céljaink megvalósításában. Ezek eléréséhez rendszerünket további kiegészítésekkel láthatjuk el, de most ennyi elég is, hiszen a témának külön szakaszt szenteltünk.

Az alaprendszer kialakítása

Még nem találkoztam olyan leírással, mely összefoglalná, hova helyezzük a lenyomatokat, a futtatható fájlokat, a lefordított kódokat, és a parancsfájlokat, vagyis hogyan csomagoljuk össze a rendszert. Ez nem csoda, hiszen ahány rendszer, annyiféle csomagolási módszer bizonyulhat megfelelőnek. Mindeközben természetesen az adott elemek elkészítése nem különbözik. A legegyszerűbb és leggyakrabban alkalmazott csomagolási módszer a hajlékonylemezek használata. Ennek általános lépései a következőkben foglalhatók össze:

1. Készítsük el a rendszerhez szükséges alkotóelemeket, így a rendszermag lenyomatát, a könyvtárakat, a parancsfájlokat, a beállításokat és minden más szükséges összetevőt.
2. Hozzuk létre az alaprendszer kezdeti fájlrendszerének könyvtárszerkezetét.
3. Helyezzük el az összetevőket a fájlrendszerben, és hozzuk létre a létfontosságú elemeket, köztük az eszközöket képviselő leírókat (device nodes) is.
4. Tömörítsük a fájlrendszert.
5. Adjuk meg a rendszermag számára, hol találhatja meg a kezdeti fájlrendszer lenyomatát, ehhez állítsuk be a megfelelő jelzőket a rendszermag lenyomatában.
6. Írjuk fel a rendszermagot és a tömörített főlenyomatot egy hajlékonylemezre, és tegyük azt indíthatóvá.

Az eljárás részletesebb bemutatására írtunk néhány fordításvezérlő fájlt, melyek gyakorlatilag a fenti lépések megvalósításához tartalmaznak útmutatásokat, és kis alaprendszert hoznak létre ingyenesen elérhető programcsomagok segítségével. Egy egyszerű – szintén ingyen letölthető – alkalmazást futtatunk, mely a netperf sever nevet kapta, feladata pedig a TCP/IP vermek teljesítményének ellenőrzése. A Netper honlapján olvashatunk a rendkívül hasznos programról (➲ http://www.nepterf.org).

Az alkalmazások indítása az alaprendszer beállításaitól függően sokféleképpen történhet. Az esetek többségében a Linux rendszermag egy indítási parancsfájlt vagy egy futtatható fájlt (init, illetve linuxrc) indít el a kezdeti fájlrendszerben a rendszerindításkor. Az init program teendői közé tartozik általában az alapfájlrendszer újrafűzése, az olvasás/írás engedélyek beállítása, más fájlrendszerek – így a proc – befűzése, valamint a rendszer más részeinek – köztük a bejelentkezési felület – üzembe helyezése, vagy közvetlenül az alkalmazás indítása. A SysVInit közismerten jól kezeli ezt a feladatot.

Alaprendszerünk esetében a bemutatáshoz nincs szükségünk bonyolult rendszerindítási folyamatra, csak egy egyszerű héjprogramot alkalmazunk, melyet bárki kedve szerint módosíthat vagy kiegészíthet:

```
mount -n -o remount,rw /
mount /proc /proc -t proc
echo  MyCompanyName, Version X.Y. Built Z,
➥ August 2000
exec /bin/sh
```

A hazai HOGYAN-ok központi oldala

Feladatként próbáljuk az echo sor tartalmát áthelyezni alkalmazásunkba, melyet a héj helyett rögtön el is indíthatunk. Az adatok kiíratását C++-ban a következőképpen oldhatjuk meg:

```
cout << COMPANY << VERSION_NO << BUILD_NO
➥ << __DATE__ << __TIME__;
```

Esetünkben a kiíratás után visszakapjuk a parancssort:

```
pipe-elinux> MyCompanyName, Version X.Y,
➥ Build Z, August 2000
pipe-elinux>
```

További kiegészítések

Ha elkészültünk az alaprendszerrel, elgondolkodhatunk azon, hogy miként tegyük gazdagabbá néhány érdekes alkalmazással. Mivel alaprendszerről van szó, itt is lépésről lépésre haladva kell kiépítenünk álmaink programbirodalmát. A következőkben felsorolt lehetőségeket mindenesetre érdemes megfontolnunk:

- egy rendszerfelépítő program: a SysVInit nagyszerű választás, azonban egyszerű alkalmazások számára kissé túlméretezett,
- egy biztonsági lehetőség: bejelentkezések támogatása,
- egy szerkesztőprogram: vi vagy emacs.
- további hálózati lehetőségek: telnet vagy ftp démonok,
- grafikus felület: X,
- nem felejtő tároló: flashmemória és merevlemez,
- további betölthető modulok,
- csomagkezelés az rpm-mel.

Valójában nincs szükség minden csomag újbóli lefordítására, ugyanis könnyen találhatunk az adott processzorhoz már lefordított változatot. Esetünkben például, ahol a cél- és a gazdagép azonos típusú, egyszerűen felhasználhatjuk a gazdagépen található lefordított fájlokat. Vegyük például a top segédprogramot – csak átmásoljuk a futtatható fájlt az alaprendszerbe, és más teendőnk nincs is. A helyzet nem minden esetben ilyen egyszerű, hiszen a helyes működéshez fel kell kutatnunk a futtatható fájl függőségeit, vagyis azokat a könyvtárakat és beállításfájlokat, amelyekre szüksége van a működéshez, de ezeket a program futtatásáig nem igazán tudjuk azonosítani. Azért itt is akad segítség, az ldd és az strace segítségével felderíthetjük a függőségeket. Nekem például csak át kellett másolnom az Emacs futtatható fájlját (emacs-nox), néhány osztott könyvtárat és beállítási fájlt az alaprendszerre, és láss csodát – a szövegszerkesztő működött. Ha nem szeretnénk mágneslemezről indítani a rendszert, egy kis pluszmunkával a következő eszközökről is megtehetjük:

- EPROM-ról,
- hálózaton keresztül,
- különböző eszközökről, így flashmemóriából, egy lemezről, CD-ről vagy ziplemezekről, vagyis a hajlékonylemez helyett szinte bármiről.
- másik operációs rendszerből.

Kapcsolódó címek

Linux HOWTO-k:
A rendszermag, a ramdisk, a gcc, a glibc, az amdisk és más HOWTO fájlok letölthetők a következő címről:
➲ http://metalab.unc.edu/pub/Linux/docs/HOWTO
➲ http://vbzo.li/linux/Magyarul-HOGYAN.html
➲ http://magyar.linux.hu

A Netperf honlapja:
➲ http://www.netperf.org/

Ha elég időt és fáradtságot szánunk rá, rendszerünket ily módon ugyanolyan gazdaggá tehetjük, mint amilyenek befutott társai.

Hibakeresés

A Linux használatának egyik nagy előnye, hogy számos segédeszköz és leírás áll rendelkezésre rendszerünk testreszabásához és gondjaink megoldásához. Beleláthatunk minden program kódjába, és a rendszer semmit sem titkol el előlünk. Emellett számos hasznos forrásmű látott napvilágot mind nyomtatásban, mind pedig a világhálón. Nincs még egy olyan rendszer, mely e tekintetben versenybe szállhatna a Linuxszal – még a FreeBSD sem.
Ha egy nehézséggel állunk szemben, általában több megoldási módozatot is kitalálhatunk. Természetesen mindig a legjobbat szeretnénk választani, azonban hogy melyik bizonyul annak, azt legtöbbször csak a kipróbálás után tudjuk eldönteni. Lapozzuk fel a megfelelő Linux HOWTO-kat és leírásokat, vagy kérdezzük meg linuxos ismerőseinket. Feladhatjuk kérdésünket egy linuxos hírcsoportban is, ahol valaki talán kisegít egy gyors válasszal.
Ha pedig mindenképpen laposabbá szeretnénk tenni pénztárcánkat, fordulhatunk olyan Linuxszal foglalkozó cégekhez is, melyek ilyen kérdések megválaszolására hivatottak. A gdb nagyszerű hibakeresési eszköz az alaprendszer alkalmazásaihoz. Ha nem tudjuk a teljes gdb programcsomagot futtatni célrendszerünkön, távolról is üzemeltethetjük kisebb részeit, akár egy gdb csonk, akár egy gdb kiszolgáló futtatásával. Végezetül, a hibakeresést nagymértékben segítheti a syslogd démon is.
A hibák felderítésére számos jól bevált módszer létezik: mindegy, melyiket használjuk – a cél a megfelelő kiút megtalálása. Általában egy sikeres példából tanulhatunk a legbiztonságosabban.
Ha belekukkantunk a Red Hat vészrendszerébe, érdekes dolgokat fedezhetünk fel. Ehhez csak a következő parancsokat kell alkalmaznunk:

```
cat rescue.img | gzip -d > rescue_root.img
mkdir rescue_root
mount -o loop rescue_root.img rescue_root
```

A rescue.img az `images` könyvtárában található tömörített lemezlenyomat. Tartalmát a következőképpen tekinthetjük meg:

```
ls rescue_root
```

Az eredmény a következő:

```
bin dev etc lib lost+found mnt proc sbin tmp usr
```

Összefoglalás

Ez az írás csupán bevezetőként szolgál az alaprendszer testreszabásához. A valódi esetekben a folyamat ennél jóval összetettebb lehet. Különösen, ha a kód szintjén is szükség van módosításokra, például egyedi eszközök támogatásánál. Mindenesetre azt láthattuk, hogy a feladat végrehajtása semmiképpen sem ördöngösség, és éppen ennek bemutatása volt a célom. Saját testre szabott alaprendszerünkkel pedig immár biztosabban haladhatunk célkitűzéseink megvalósítása felé.

He Zhu
(hezhu@yahoo.com)
a rendszerprogramok és a hálózatok terén otthonos. Jelenleg New Jersey-ben dolgozik, a Bell Labs kötelékében.