Linux rendszergazda tanfolyam

III. Mélyvíz

Lilo boot manager

- Boot folyamat ismétlés
- Fő feladata a kiválasztott operációs rendszer elindítása. Linux esetén a megfelelő kernel betöltése.
- A Lilo boot manager elhelyezhető az MBR-en kívül, a boot-szektorban is, vagyis akár más boot manager-ekkel is együtt tud működni.
- Konfigurációs állománya az */etc/lilo.conf*.
- A lilo használható más operációs rendszerek indítására is.
- Képes több Linux disztribúció kezelésére.
- Egy disztribúción belül képes több különböző kernel image indítására.
- Lehetőség van menüs indításra is. A menüben a kurzorbillentyűk és az <enter> segítségével választhatunk.
- Időzített automatikus rendszerindítást is ismer.

Lilo konfigurálás

- Példa
- A /etc/lilo.conf fájl egy egszerű szövegfájl, mint általában minden konfigurációs fájl a Linuxban, így bármely editorral szerkeszthető.
- A „prompt” paraméter esetén indulásnál egy menü jelenik meg, egyébként semmi sem, csak elindul az alapértelmezett (default) rendszer.
- Az „image” kulcsszóval minden esetben egy Linux lefordított kernel image fájlt határozunk meg. **Ha itt hibázunk, a rendszer nem fog tudni elindulni!** => Ajánlatos külön biztonsági indítólemezt készíteni már a telepítés során!
- Az „other” kulcsszó nem Linuxos rendszerek indítására való. DOS, illetve Windows-os rendszerek esetén azonban, szükség van egy külön betöltő program („chain loader”) helyének megadására is.
- **A konfiguráció módosítása után *root*-ként mindig ki kell adni a *„lilo”* parancsot!**

A Lilo további lehetőségei

- A Lilo készenléti jelénél a <Tab> billentyű lenyomása megmutatja a választási lehetőségeket. Begépelve a választott rendszer nevét, elindul a rendszer.
- Ha a Lilo-t nem interaktív módba állították be (nincs prompt paraméter), A *„Lilo”* üzenet megjelenése előtt már nyomva kell tartani az <Alt> vagy a <Shift> billentyűket. Ezután már használható a <Tab> billentyű is.
- A Lilo telepítése gyakorlatilag a *„lilo”* parancs kiadását jelenti. Legelső esetben lementi egy fájlba azt a szektort, amit majd felül fog írni (MBR vagy valamely boot szektor). MBR-be történő telepítés esetén ez a */boot/boot.0300* nevű fájl.
- A Lilo eltávolítása az eredeti állapot visszaállítását jelenti, melyet a következő paranccsal lehet elvégezni: *„dd if=/boot/boot.0300 of=/dev/hda bs=446 count=1”*. Vagy DOS alól az *„fdisk /mbr”* parancs is törli a Lilo-t.

A kernel elindulási folyamata

- A Linux rendszermag a lemezen tömörítve van, ezért először kicsomagolja önmagát.
- Első feladata a videókártya inicializálása. Ha úgy állítottuk be, akkor kérheti a képernyőfelbontás megadását, ellenkező esetben a kernelben meghatározott videómódot állítja be a videókártyán.
- Ezután a rendszermag ellenőrzi, milyen hardver elemek (merevlemezek, hajlékonylemez meghajtók, hálózati kártyák, stb.) léteznek a gépben, és megpróbálja ezek eszközmeghajtóit megfelelően beállítani.
- Ezután megpróbálja felcsatolni (*mount*) a gyökér fájlrendszert (*root* partíció - /) *read-only* módban.
- Az esetleges kernel modulok betöltése következik a */etc/modules* fájl alapján.
- Ezután elindítja az *„init”* (*/sbin/init*) programot. A további műveleteket már az *„init”* fogja végrehajtani. A rendszer legelső programja az *„init”*, így övé az *„1”*-es processz azonosító!

A „*init*” feladatai

- Alapfeladata a rendszer inicializálása.
- Konfigurációs állománya a */etc/inittab* fájl.
- Elindítja a */etc/rcS.d* könyvtárba linkelt programokat.
- A továbbiak a beállított futási szinttől függenek. A futási szint beállítható a Lilo promptjánál, és az *inittab*-ban is. Végrehajtja azokat a programokat, amelyekre a */etc/rcX.d* könyvtárban lévő linkek mutatnak. Itt az „*X*” magát a futási szintet jelöli (0-tól 6-ig egy szám).
- Beállítja, hogy mit kell tennie a rendszernek a Ctrl-Alt-Del billentyűk lenyomása esetén.
- Ha rendelkezünk UPS-sel, akkor itt meghatározhatjuk, hogy mi történjen különböző események alkalmával.
- Ezután elindítja a beállított *getty* programokat a meghatározott terminálokra - ezek teszik lehetővé a bejelentkezést a rendszerbe. Lehetőségünk van a soros portokra is különböző *getty* programokat indítani.

Futási szintek

Szint	Leírása
0	Rendszerleállítás (*halt*)
1	Egyfelhasználós (*single-user*) mód rendszeradminisztrációhoz
2	Többfelhasználós mód parancssoros bejelentkezéssel
3	Többfelhasználós mód parancssoros bejelentkezéssel
4	Nem használt
5	Többfelhasználós mód grafikus bejelentkezéssel
6	Újraindítás (*reboot*)

- Futási szintet az „*init*” paranccsal válthatunk úgy, hogy megadjuk a kívánt futási szint számát paraméterként. Ezért lehet leállítani a rendszert az „*init 0*” paranccsal.
- Az „1”-es szint csak speciális feladatok elvégzésekor használatos. Ilyen például a fájlrendszerbeli hibák javítása.
- Van olyan disztribúció, ami a fentiektől eltérő futási szint értelmezéseket használ.

A „/etc/*inittab*” lehetőségei

```
id:2:initdefault:               # alapértelmezett futási szint
si::sysinit:/etc/init.d/rcS     # boot-oláskor végrehajtandó scriptek
~~:S:wait:/sbin/sulogin         # mi történjen single-user módban
l0:0:wait:/etc/init.d/rc 0      # 0-s szint esetén mi történjen
l1:1:wait:/etc/init.d/rc 1      # 1     -   "   -
l2:2:wait:/etc/init.d/rc 2      # 2
l3:3:wait:/etc/init.d/rc 3      # 3
l4:4:wait:/etc/init.d/rc 4      # 4
l5:5:wait:/etc/init.d/rc 5      # 5
l6:6:wait:/etc/init.d/rc 6      # 6
z6:6:respawn:/sbin/sulogin      # normál esetben nem elérheto vészállapot
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now   # alt+ctrl+del beállítás
# Konzolok megadásának formája:  <id>:<runlevels>:<action>:<process>
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
# Szünetmentes (UPS) esetén mi történjen az egyes események esetén
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
#T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100  # Soros portokon elérheto konzol
```

Amint látszik az egyes futási szintekhez tartozó indító script-eket a */etc/init.d/rc* script fogja indítani. De ehelyett mondhatunk mást is, más kérdés, hogy nem ajánlott! Szintén módosíthatjuk a konzolok számát, sőt még azt is, hogy az egyes konzoloknál milyen program fusson!

Az „*rc*” script-ek 1

- Azért nevezik ezeket a programokat script-eknek, mert minden esetben *shell script*-ekről van szó.
- Debian esetén ezek a script-ek a */etc/init.d* könyvtárban vannak.
- A script-ek rendszerint négy paramétert kezelnek:
  - *start* - indítás
  - *stop* - leállítás
  - *reload* – újratöltés, rendszerint újrakonfigurálás után
  - *restart* - újraindítás
- Mi is készíthetünk ilyen indító script-eket, saját programjainknak.
- A script-ek helye és formája disztribúciótól is függ!
- A céljük azonban közös. Szolgáltatások (démonok) indítása, leállítása és újraindítása. Ebből adódik, hogy bármikor kiadhatók ezek a parancsok a rendszerben. A *„/etc/apache stop”* parancs például leállítja a web szervert, ha az futott.

Az „*rc*” script-ek 2

- A /etc/rcX.d könyvtárakban lévő linkek elnevezési szabályai:
  - Az első betű vagy „*S*” vagy „*K*”. „*S*” esetén az adott script *start* paraméterrel lesz indítva, „*K*” esetén pedig *stop* paraméterrel.
  - A következő szám azt határozza meg, hogy hányadikként induljon el az adott script. Minél kisebb, annál hamarabb indul. Több script is kaphatja ugyanazt a számot, ez nem okoz semmilyen problémát.
  - A végén rendszerint ugyanaz a név szerepel, mint amire mutat az */etc/init.d* könyvtárban.
- Ezek szimbolikus linkek, létrehozásuk az „*ln –s*” paranccsal történik. Pl. a „*ln –s ../init.d/ts2 S99ts2*” parancs létrehoz egy linket a *ts2* indító script-hez relatív hivatkozással, ami az utolsók között (*99*) fog elindulni („*S*”).
- Bármikor törölhetők. Feltéve, hogy tényleg nem szükséges a rendszer normál működéséhez.
- A linkeket minden futási szinten külön kell létrehozni!

Eszközkezelés

- Telepítés. Általában minden eszközfájl rendelkezésünkre áll a */dev* könyvtárban. Ha valami miatt mégis hiányzik egy eszközfájl, akkor azt nekünk kell létrehoznunk. Erre két lehetőségünk van:
  - a */dev* könyvtárban található *MAKEDEV* script segítségével vagy az – Pl: *„/dev/MAKEDEV -v ttyS0”*
  - *mknod* parancs segítségével
- Az aleszköz- és a főeszköz-szám ,,kitalálásában'' a */usr/src/linux/Documentation/device.txt* fájl segít.
- Az eszközöknek egy listája olvastó a *Linux rendszeradminisztrátorok kézikönyv* 5. fejezetében.
- A lemezek adminisztrálásának alapvető feladatai:
  - Merevlemez partícionálás
  - Formázás
  - Fájlrendszer készítés az egyes partíciókon
  - A különböző fájlrendszerek csatolása (*mount*) automatikusan vagy kézzel. Szükség lehet a lecsatolásukra (*umount*) is.

Partícionálás

- **Óvatosan!** Már létező rendszerek esetén, a partícionálás előtt ajánlott mentéseket végezni!
- Linux alatt partícionálásara az *fdisk* és a *cfdisk* parancs használható. A *cfdisk*-nek kezelhetőbb felületete van, de mindkettővel minden (létrehozás, törlés, aktív partíció kiválasztása) elvégezhető. **Minkét esetben lényeges, hogy az esetleges módosítások elvégzése után az eredményt vissza kell írni (*write*) az eszközre!**
- A partíciók listája az „fdisk –l” paranccsal jeleníthető meg.
- Az *fdisk* használatához, paraméterül meg kell adnunk az eszköz nevét is (pl.: „*fdisk /dev/hda*”)
- A partíciók létrehozásakor a típusát is meg kell határozni, ami egy szám. A partíciók típusa az *fdisk* paranccsal megjelenítve: partíció típusok.
- Az elsődleges (*primary*) partíciók az eszközfájlok tekintetében 1-től 4-ig sorszámozódnak, míg a logikai partíciók 5-től kezdődnek. Ha tehát például van egy *primary* partíciónk és van egy logikai partíciónk, akkor a logikai partíció eszközfájljának a neve a */dev/hda5*.
- A partíciók átméretezésére is van mód, de ahhoz már más programokat kell használnunk. Például a *parted* már egy egész jól használható partícionáló program.

Partícionálási sémák

- Nem könnyű egy lemezt a lehető legjobb módon partícionálni. Nincs erre univerzális repect, mivel túl sok tényezőt kell figyelembe venni.
- A szokásos mód az, hogy egy viszonylag kicsi gyökér fájlrendszert hozunk létre, mely a */bin*, */etc*, */dev*, */lib*, */tmp* könyvtárakat és olyan fájlokat tartalmaz, melyek a rendszer felállásához és futtatásához szükségesek.
- A többi fontos rész külön partíción helyezkedik el, azaz a */usr*-nek, */home*-nak (a felhasználók saját könyvtárai) és a swap területnek külön partíciót tartunk fenn.
- Külön partícióra szokás tenni még a */var* és /tmp tartalmát is.
- A sok partícióval az a probléma, hogy a teljes lemezterületet több kis részre osztja. Egy telepített rendszer esetén viszont utólag már nem módosíthatunk a partícióméreteken, ha kiderül pl. hogy több kellene a */home*-nak, de kevesebb is elég lenne a */usr*-nek. Manapság, amikor a lemezek és az operációs rendszerek egyre megbízhatóbbak, sokan inkább egyetlen nagy partíciót hoznak létre, mely az összes fájlt tartalmazza. Ez viszont megnehezíti a biztonsági mentéseket és több szempontból rontja a rendszer megbízhatóságát.

Formázás

- A *formázás* az a folyamat, melynek során a mágneses adathordozóra jeleket írunk, melyek a sávokat és szektorokat jelölik meg. Formázás nélkül nem használhatók a lemezek.
- A hajlékonylemezeket az *fdformat* paranccsal formázhatjuk. A megadott hajlékonylemez eszközfájl az egyetlen paraméter. Például a következő parancs egy 3,5 hüvelykes HD hajlékonylemezt formáz meg az első meghajtóban: „*fdformat /dev/fd0H1440*”. Itt a „H” a HD jelölése, az utána lévő szám pedig a méretre utal (1,44Mb).
- Hiba esetén az *fdformat* csak jelzi, hogy volt valami probléma, de nem mondja meg a helyét. A *badblocks* parancs használható a hibák helyének megjelenítésére.
- Merevlemezek esetén a formázás nem szükséges!

Fájlrendszer készítés

- *Fájlrendszer*en (*filesystem*) azokat a módszereket és adatstruktúrákat értjük, melyeket egy operációs rendszer használ egy lemez vagy partíció fájljainak kezelésére, azaz ahogyan a fájlok elrendeződnek a lemezen.
- Mielőtt egy lemezt vagy partíciót fájlrendszerként kezdünk használni, inicializálni kell, és a nyilvántartó adatstruktúrákat a lemezre kell írni. Ezt a folyamatot *fájlrendszer készítés*nek nevezzük.
- A fájlrendszerek létrehozása (inicializálása) az *mkfs* paranccsal történik. Tulajdonképpen minden fájlrendszer típushoz különálló program létezik, és az *mkfs* parancs csak egy előtét, mely ezek közül a megfelelőt futtatja. A típust a „*-t fstype*” (fájlrendszer típusa) opcióval választhatjuk ki.
- Például hajlékonylemez formázása hibaellen-őrzéssel az „*mkfs -t ext2 -c /dev/fd0H1440*” paranccsal történhet. A lemezen létrejövő fájlrendszer *ext2* típusú lesz!

Fel- és lecsatolás

- A fájlrendszerek használatba vétele előtt fel kell *csatolni* őket. Mivel a UNIX-ban minden fájl egyetlen könyvtárstruktúrába illeszkedik, a felcsatolás művelete az új fájlrendszer tartalmát úgy jeleníti meg, mintha alkönyvtár lenne egy már felcsatolt fájlrendszerben.
- Fájlrendszer felcsatolására a *mount*, lecsatolására pedig az *umount* parancs használható.
- A *mount* parancsot rendszerint két paraméterrel hívjuk meg, az első az eszközfájl neve, a második pedig annak az alkönyvtárnak az útvonala, ami alá be szeretnénk csatolni. A könyvtárnak már léteznie kell a parancs kiadása előtt! Például: „*mount /dev/hda5 /home*” hatására a *hda* első logikai egysége a */home* könyvtár alatt lesz elérhető.
- Azt a könyvtárat, ahová egy fájlrendszer becsatolásra kerül, csatolási pontnak (*mount point*) nevezik.
- A fájlrendszer típusát is megadhatjuk „*-t fstype*” formában, de a *mount* parancs igen sok fájlrendszert automatikusan felismer.
- A „*-r*” paraméter hatására csak olvashatóként (*read-only*) kerül felcsatolásra a fájlrendszer.
- Az *umount* parancsnak elég csak az egyik adat (eszközfájl vagy csatolási pont) a leválasztáshoz. Például az „*umount /dev/hda5*” ugyanazt végzi el, mint az „*umount /home*”.

Az /etc/fstab szerepe

- Ebben a fájlban lehet elhelyezni az állandó fájlrendszerek elérhetőségét, csatolási pontjaikat, fájlrendszerüket, és elérési sajátosságaikat.
- Egy példa fstab fájl:

```
# <file system> <mount point>   <type>  <options>              <dump>  <pass>
/dev/hda2       /               ext3    errors=remount-ro      0       1
/dev/hda1       none            swap    sw                     0       0
proc            /proc           proc    defaults               0       0
/dev/fd0        /floppy         auto    user,noauto            0       0
/dev/cdrom      /cdrom          iso9660 ro,user,noauto         0       0
```

- A *mount* parancs is innen olvassa ki az adatokat. Így ha a cdrom-ot szeretnénk használni, elég csak a „*mount /cdrom*” parancsot használni.
- Ráadásul az *fstab*-ban lévő opciók lehetővé teszik, hogy egy mezei felhasználó is probléma nélkül *mount*-olhasson fel eszközöket. Egyébként ehhez *root* jogokra lenne szükség! Az options részben szereplő „*user*” opció azt jelenti, hogy az adott fájlrendszert egy egyszerű felhasználói is felcsatolhatja.
- Az „*ro*” opció hatására pedig *read-only*-ként kerül felcsatolásra az adott fájlrendszer. Itt a példában a CD meghajtónál találkozhatunk vele.
- A „*noauto*” jelzés azt mondja meg a rendszernek, hogy ha képes is lenne az automatikus felcsatolásra, akkor se alkalmazza. Cdrom esetén egyébként képes lenne erre!

Fájlrendszer ellenőrzés - fsck

- Egy fájlrendszer helyessége és érvényessége az **fsck** paranccsal ellenőrizhető. Ez a program képes a megtalált kisebb problémákat kijavítani, és figyelmeztetni a nem javítható hibákra.
- A legtöbb rendszer úgy van beállítva, hogy rendszerindításkor automatikusan futtatja az **fsck** programot (*fstab pass* oszlopban lévő 1-es érték hatására), így remélhetően minden hiba kiderül és javításra kerül a rendszer használata előtt.
- **Az fsck programot csak lecsatolt fájlrendszeren szabad alkalmazni!**
- Ha az **fsck** javíthatatlan hibát észlel, akkor vagy nagyon alapos tudásra van szükség a fájlrendszer működését illetően, vagy jó biztonsági mentésre.
- Lemezhibák ellenőrzésére használható a badblocks parancs. A hibás szektorok listája külön fájlba menthető és átadható az fsck parancsnak, hogy az operációs rendszer a későbbiekben ne használja a hibás lemezterületeket. Például egy floppy esetén:

  „badblocks /dev/fd0H1440 1440 > bad-blocks”

  „fsck -t ext2 -l bad-blocks /dev/fd0H1440”

Swap

- A Linux a fájlrendszer egy közönséges fájlját vagy egy külön partíciót is tud használni *swap* területként. A *swap* partíció gyorsabb, viszont a *swap* fájl mérete sokkal könnyebben állítható.
- A Linux megengedi egyszerre több *swap* partíció és/vagy fájl egyidejű használatát. Ez azt jelenti, hogy ha esetenként szokatlanul nagy *swap* területre van szükség, egy új *swap* fájl létrehozásával megoldható a probléma a teljes terület állandó lefoglalása helyett.
- A *swap* fájl egy közönséges fájl; a rendszermag sem kezeli speciális módon. Az egyetlen, ami számít a rendszermagnak, hogy ez a fájl ne tartalmazzon lyukakat, és elő legyen készítve a *mkswap* paranccsal. Mindenképpen helyi lemezen kell lennie. Létrehozása: *„dd if=/dev/zero of=/extra-swap bs=1024 count=1024”*, ahol */extra-swap* a *swap* fájl neve, a mérete pedig a *count*= után van megadva (jelen példánkban kilobyte-ban).
- A *swap* partíció egy egyszerű fájlrendszer nélküli partíció. Jó, ha a *swap* partíció 82-es típusú (*Linux swap*), ez világosabbá teszi a partíciós táblát, de a *kernel* nem figyeli a partíció típusát.

Rendszernaplók - logok

- A rendszerindulás folyamán két fontos démon indul el, ami a rendszer üzeneteket naplózza (logolja), a *klogd* és a *sysklogd*. A *klogd* a kernel üzeneteit naplózza, míg a *sysklogd* minden rendszer üzenetet naplóz.
- A naplófájlok helye a */var/log* könyvtár. A *klogd* a *kern.log* fájlba dolgozik, a *sysklogd* pedig a *syslog* fájlba.
- Ugyancsak a */var/log* könyvtárban kerülnek tárolásra a bejelentkezések adatai (*auth.log*, *lastlog*, *user.log*), a démonok üzenetei (*daemon.log*), és rendszerint ebben a könyvtárban létrehozott könyvtárakban naplóznak a külön telepített különböző szolgáltatások is (web, ftp, dns, stb).
- Állandó folyamatos működés esetén, ezek a *log* fájlok igen nagyra is nőhetnek! A rendszer ez ellen úgy védekezik, hogy rendszere időközönként (naponta, hetente, havonta) rotálja a fájlokat, átnevezi a régit, tömöríti és létrehoz egy újat. Ezért lehet találni a rendszerben például *syslog.0*, *syslog.1.gz*, stb fájlokat is. Ezt a rotálást a *logrotate* parancs végzi, amely természetesen külön konfigurálható (*/etc/logrotate.conf*).

Kernel és kernel modulok

- Ha egy adott eszközt szeretnénk használni, akkor a kernelnek is támogatnia kell. Vagy be kell lennie fordítva a kernelbe, növelve annak méretét, vagy külön modulként kell betölteni, amikor szükség van rá. Magyarul Linux esetén a driverek (eszközkezelő programok) a kernelhez kapcsolódnak, vagy közvetlenül a kernelbe égetve, vagy külön betölthető modulként.
- Minél újabb a kernel, annál több eszközt támogat. De előfordulhat olyan is, hogy az adott eszközt linux alatt nem lehet munkára bírni, mert nincs hozzá kezelőprogram.
- A modulokat külön kell lefordítani, és szinte bármikor be lehet illeszteni a futó kernelbe, illetve ki lehet venni onnan.
- Lényeges, hogy a 2.6-os sorozatú kernelek esetén már más a modulkezelés, így a régebbi modulok nem használhatók közvetlenül!