A bekapcsolástól a bash promptig

Greg O'Keefe, gcokeefe@postoffice.utas.edu.au v0.9, 2000 november

Ez a dokumentum röviden leírja, hogy mi történik egy Linux rendszerben a gép bekapcsolásától a bash prompt megjelenéséig. Ezek megértése hasznos lehet, amikor problémákat kell megoldanod vagy be kell állítanod a rendszert.

Contents

1 Bevezetés

Idegesítőnek tartottam, hogy sok olyan dolog történik a Linux gépem belsejében, amit nem értek. Ha Te is, úgy mint én, szeretnéd igazán megérteni a rendszered működését ahelyett, hogy csak a használatát tanulod

meg, ez a dokumentum egy jó kiindulási pont lehet. Ez a fajta háttérismeret kellhet akkor is, ha elsőrangú linuxos problémamegoldó szeretnél lenni.

Feltételezem, hogy van egy működő Linux géped, és tudsz néhány alapvető dolgot a Unix-ról és a PC hardveréről. Ha nem, akkor nagyon jó kiindulási alap a tanuláshoz az Eric S. Raymond-féle

The Unix and Internet Fundamentals HOWTO <http://www.tldp.org/HOWTO/Unix-and-Internet-Fundamentals-HOWTO.html> (A Unix és az Internet alapjai HOGYAN). Ez rövid, nagyon olvasmányos doksi és lefedi az alapokat.

A dokumentum fő irányvonala az, hogy a Linux hogyan indul el. Ezen felül egy részletesebb tanulmányi forrás próbál lenni. Minden részhez mellékeltem gyakorlatokat is. Ha ténylegesen megcsinálsz néhányat közülük, sokkal többet tanulsz, mint csak az olvasással.

Remélem, néhány olvasó vállalkozik a tudomásom szerinti legjobb, a Linux megismerését segítő feladatra, ami a rendszer forráskódból való felépítése. Az olasz filozófus, Giambattista Vico (1668-1744) szerint "verum ipsum factum", vagyis "a dolgokat készítésük közben értjük meg igazán". Köszönet Alexnek (lásd 14.5 (Köszönetnyilvánítások)) az idézetért.

Ha el szeretnéd készíteni a saját Linuxodat, elolvashatod még Gerard Beekman

Linux From Scratch HOWTO <http://www.linuxfromscratch.org> -ját (Linux a kezdetektől HOGYAN, LFS). Ez részletes útbaigazítást ad egy teljes, használható rendszer forráskódból való felépítéséről. A LFS weboldalán levelezőlistát is találsz, ami azoknak szól, akik ezen a módon építenek fel rendszereket. Néhány instrukció, ami régebben része volt ennek a dokumentumnak, most már egy másik leírásban található meg, címe "Minimális Linux rendszer építése forráskódból", és itt található meg:

From PowerUp to Bash Prompt honlapja <http://www.netspace.net.au/~gok/power2bash/> . Elmagyarázzák, hogyan "játsszunk" a rendszerrel, tisztán mint tanulási gyakorlat.

A csomagok abban a sorrendben jelennek meg, amelyikben a rendszer indulási folyamatában is részt vesznek. Ez azt jelenti, hogy ha ebben a sorrendben telepíted fel őket, újraindítást csinálhatsz minden telepítés után, és láthatod, hogy a rendszer minden egyes alkalommal közelebb visz ahhoz, hogy megkapd a bash prompt-ot. Van ebben egy bizonyosságot erősítő érzés.

Azt ajánlom, hogy először olvasd el a részek fő szövegeit, átugorva a gyakorlatokat és hivatkozásokat. Ezután döntsd el, milyen mélységű szakértelmet akarsz elérni, és mennyi erőfeszítést vagy hajlandó rááldozni. Ezután kezdd el megint az elejétől, végigcsinálva a gyakorlatokat és a kiegészítő olvasmányokat áttanulmányozva.

2 Hardver

Amikor bekapcsolod a számítógépet, az leteszteli magát, hogy minden rendben működik-e. Ezt "bekapcsolási önteszt"-nek (POST) hívják. Ezután a bootstrap loader nevű, a ROM BIOS-ban található program keres egy bootszektort. A bootszektor egy lemez első szektora, és egy kis programkódot tartalmaz, ami képes egy operációs rendszer betöltésére. A bootszektorokat a bűvös 0xAA55=43603 számmal jelölik meg, ez a 0x1FE=510-es bájtnál található. Ez a szektor utolsó két bájtja. Ezen a módon képes eldönteni a hardver, hogy a szektor bootszektor vagy sem.

A bootstrap loader egy listával rendelkezik arról, hogy hol keressen bootszektort. Az én régi gépem először az elsődleges floppy-meghajtón kereste, majd az elsődleges merevlemezen. A modern gépek ezenkívül CD-ROM-on is kereshetik. Ha megtalálja a bootszektort, betölti a memóriába, és átadja a vezérlést az operációs rendszert betöltő programra. Egy tipikus Linux rendszeren ez a LILO első fokozatú (first stage) betöltője. Ezen kívül persze még sok más módon lehet betöltetni a rendszert. Lásd a *LILO User's Guide*-ot részletekért. Lásd még a 3.3 (LILO) részt URL-ekért.

Természetesen még rengeteg mindent lehetne mondani arról, hogy mit csinál a PC hardvere, de ez nem ennek

a helye. Erről a témáról sok jó könyvet olvashatsz .

2.1 Beállítás

A gép néhány információt tárol önmagáról a CMOS-ban. Ez magában foglalja a gépben található lemezek és memória adatait/jellemzőit. A gépben lévő BIOS tartalmaz egy programot, amivel ezeket a beállításokat módosítani lehet. Figyeld meg a géped indulásakor megjelenő üzeneteket, ezek egyike tájékoztat az ehhez valóó hozzáférés módjáról. Az én gépemen a "Delete" gombot kell megnyomni, mielőtt elkezdené betölteni az operációs rendszert.

2.2 Gyakorlatok

Nagyon jó módszere a hardverről való ismeretszerzésnek, ha használt alkatrészekből épít egy gépet az ember. Végy legalább egy 386-ost, amin már könnyűszerrel futtathatsz Linuxot. Nem fog sokba kerülni. Kérdezgess körbe, hátha valaki tud is adni olyan részegységeket, amire szükséged van.

Nézz körül a Neten, töltsd és fordítsd le, majd készíts egy indítólemezt az

Unios <http://www.netspace.net.au/~gok/resources> -nak. (Volt egy honlapjuk: <http://www.unios.org> , de ez már eltűnt.) Ez csak egy bootolható "Helló világ!" program, alig több mint 100 sornyi assembler kód. Jó volna látni olyan formátumban, hogy a GNU assembler, az "`as`" is megértse.

Nyisd meg egy hexa-editorral az Unios indítólemezének képfájlját. Ez a fájl 512 bájt méretű, pontosan egy szektornyi. Keresd meg a bűvös 0xAA55-ös számot. Tedd ugyanezt egy indítófloppy-val vagy a saját gépeddel. Használhatod a "`dd`" programot egy fájlba való kimásoláshoz: `dd if=/dev/fd0 of=boot.sector`. Légy *nagyon* óvatos, hogy az "`if`" (bemeneti fájl) és "`of`" (kimeneti fájl) jól legyen megadva!

Nézd meg a LILO betöltőjének forráskódját.

2.3 További információ

- *The Unix and Internet Fundamentals HOWTO* <http://www.tldp.org/HOWTO/Unix-and-Internet-Fundamentals-HOWTO.html> , szerző Eric S. Raymond, különösen a 3. fejezet: *What happens when you switch on a computer?* (Mi történik, ha bekapcsolod a számítógépet?)
- Az első fejezet a *The LILO User's Guide*-ból kitűnő magyarázatot ad a PC lemezpartíciókról és a bootolásról. Lásd a 3.3 (LILO) fejezetet URL-ekért.
- *Az újJ Peter Norton Programmer's Guide to the IBM PC & PS/2*, szerző Peter Norton és Richard Wilton, kiadó Microsoft Press, 1988. Van egy újabb Norton-könyv, ami jól néz ki, de még nem engedhetem meg magamnak!
- A PC-k fejlesztéséről szóló más könyvek egyike

3 LILO

Amikor a számítógép betölt egy bootszektort egy normál Linux rendszeren, valójában a LILO egy részé tölti be, amit "első fokozatú betöltőnek" hívnak. Ez egy kicsi program, aminek nincs más szerepe, mint betölteni és futtatni a "második fokozatú betöltőt".

A második fokozatú betöltő ad egy készenléti jelet (ha úgy állították be) és betölti az általad választott operációs rendszert.

Amikor a rendszer elindult, működik, és futtatod a `lilo`-t, akkor valójában a "map installer"-t futtatod. Ez kiolvassa a `/etc/lilo.conf` konfigurációs fájlt, kiírja a betöltőket és az információkat az általa betölthető operációs rendszerekről a merevlemezre.

Rengeteg egyéb módja van még a rendszered betöltésének. Amit leírtam, a legkézenfekvőbb és "normál" mód, legalábbis egy Linuxot, mint fő operációs rendszert futtató gép számára. A LILO felhasználói kézikönyv többféle példát sorol fel a "betöltési koncepciókról". Megéri elolvasni, és néhányat kipróbálni közülük.

3.1 Beállítás

A LILO konfigurációs fájlja a `/etc/lilo.conf`. Létezik kézikönyv-oldala: gépeld be a `man lilo.conf` parancsot a megtekintéséhez. A fő dolog a `lilo.conf`-ban egy bejegyzés mindenhez, amit a LILO be tud tölteni. A Linux esetében ez tartalmazza a kernel helyét, és hogy melyik lemezpartíciót kell gyökér fájlrendszerként befűzni. Más operációs rendszereknél a fő információ az, hogy melyik partícióról bootoljanak.

3.2 Gyakorlatok

VIGYÁZAT: nagyon vigyázz ezekkel a gyakorlatokkal. Nagyon könnyű elrontani valamit, tönkretenni a master boot rekordot és használhatatlanná tenni a rendszert. Bizonyosodj meg, hogy van egy működő mentőlemezed, és tudod is használni arra, hogy visszaállítsd a dolgokat. Lásd alább a hivatkozást a tomsrtbt-re, arra a mentőlemezre amit én használok és ajánlok. A legjobb az, ha olyan gépet használsz amelyiknél mindez nem számít.

Telepítsd a LILO-t egy floppy-ra. Nem számít a nincs más rajta, csak egy kernel - kapsz majd egy "kernel pánik"-ot, amikor készen áll az init betöltésére, de legalább tudod, hogy a LILO működik.

Ha akarod, még tovább léphetsz és megnézheted, mekkora rendszert tudsz felrakni a floppy-ra. Ez valószínűleg a második legjobb Linux-tanulási tevékenység. Lásd a Bootdisk HOWTO-t (Indítólemez HOGYAN, URL alább) és a tomsrtbt-t (URL alább) tippekért.

Tölsd be az Unios-t a LILO-val (lásd a 2.2 (hardver gyakorlatok) részt URL-ekért). Extra gyakorlatként nézd meg, végre tudod-e hajtani ezt egy floppy-lemezen.

Készíts egy boot-ciklust. Töltesd be a master boot rekord-beli LILO-val az egyik elsődleges partícióban lévű boot szektort, majd azzal újra a MBR-beli LILO-t... Esetleg használd a MBR-t és mind a négy elsődleges partíciót egy öt lépéses ciklushoz. Süti!

3.3 További információ

- A LILO kézikönyv-oldala
- A LILO csomag (*lilo* <ftp://lrcftp.epfl.ch/pub/linux/local/lilo/>), ami tartalmazza a "LILO User's Guide"-ot is `lilo-u-21.ps.gz` (vagy egy újabb verzió). Talán már megvan neked ez a dokumentum. Nézd meg a `/usr/doc/lilo` vagy valami hasonló helyen. A postscript verzió jobb, mint a sima szöveg, mivel diagramokat és táblázatokat is tartalmaz.
- A *tomsrtbt* <http://www.toms.net/rb> , a legjobb egyfloppy-s disztribúció. Nagyon jó mentőlemezként is.
- *A Bootdisk HOWTO* <http://www.tldp.org/HOWTO/Bootdisk-HOWTO/> (Indítólemez HOGYAN)

4 A Linux kernel

A kernel valójában nagyon sok mindent csinál. Úgy gondolom jó összefoglalása a dolgoknak az, hogy ráveszi a hardvert, azt tegye amit a programok akarnak, méghozzá hatékonyan és jól.

A processzor egyszerre csak egy utasítást tud végrehajtani, de a Linux rendszerek látszólag több dolgot is végeznek egyszerre. A kernel ezt úgy éri el, hogy nagyon gyorsan váltogatja a feladatokat. Ezzel a lehető legjobban ki lehet használni a processzort, úgy, hogy követjük melyik folyamat kész a futásra, és melyik várakozik valamire - mint például egy rekordra egy merevlemez fájlból, vagy egy billentyű leütésére. Ezt nevezik ütemezésnek (scheduling).

Ha egy program nem csinál semmit, nem kell a memóriában lennie. Még egy olyan programnak is ami csinál valamit, lehet, hogy vannak olyan részei amik nem csinálnak semmit. Mindegyik folyamat (process) címzési tartománya fel van osztva lapokra. A kernel nyilvántartja, melyik folyamat melyik lapja van a legtöbbet használva. Azokat a lapokat, amik kevésbé használtak, ki lehet helyezni a csere (swap) partícióra. Amikor megint szükség van rájuk, egy másik nem használt lapot lehet kihelyezni. Ez a virtuális memóriakezelés.

Ha már fordítottál saját kernelt, észrevehetted milyen sok az eszközfüggő beállítás. A kernel sok egyedi kódot tartalmaz, a különböző hardverekkel történő kommunikáláshoz, valamint hogy szép egységes felületet mutasson az alkalmazások felé.

Szintén a kernel irányítja a fájlrendszert, a folyamatok közötti kommunikációt, és egy csomó hálozati dolgot.

Amikor a kernel betöltődött, az első dolog amit csinál, hogy egy "`init`" programot keres és lefuttaja.

4.1 Beállítás

A kernel beállításának legnagyobb része akkor történik, amikor futtatod a "`make menuconfig`"-ot vagy a "`make xconfig`"-ot az `/usr/src/linux/` könyvtárban, (vagy bárhol máshol, ahol a Linux kernel forráskód van). Újra tudod állítani az alapértelmezett video módot, gyökér (root) fájlrendszert, csere (swap) eszközt és a RAM disk méretét, ha az "`rdev`"-et használod. Ezeket - vagy még több paramétert - a lilo is át tud adni a kernelnek. Adhatsz a lilo-nak paramétereket a lilo.conf fájlban, vagy a lilo promptjánál, hogy átadja ezeket az információkat. Például ha a hda3-at akarod használni mint root fájlrendszert a hda2 helyett, begépelheted hogy:

```
LILO: linux root=/dev/hda3
```

Ha forrásból építesz rendszert, sokkal egyszerűbbé teheted az életet, ha "monolitikus" kernelt csinálsz. Ez azt jelenti, hogy nincsenek moduljai. Ekkor nem kell kernel modulokat másolni a cél rendszerbe.

FIGYELEM: A `System.map` fájlt a kernel logger (naplózó) használja, hogy meghatározza a modulneveket generáló üzeneteket. A `top` nevű program szintén használja ezt az információt. Amikor a kernelt a célrendszerbe másolod, másold át a "`System.map`"-ot is.

4.2 Gyakorlatok

Egy kis gondolkodnivaló: a `/dev/hda3` egy különleges fájl, ami leír egy merevlemez particiót. De egy olyan fájlrendszeren van, mint bármelyik másik fájl. A kernel meg akarja tudni, hogy melyik particiót csatolja fel (mount), mint gyökér fájlrendszert - de még nincs is fájlrendszere. Mégis hogyan tudja olvasni a `/dev/hda3` partíciót, hogy megtudja melyik lemezrészt csatolja?

Ha még nem tetted meg, készíts saját kernelt. Olvass el minden súgóinformációt az opciókhoz.

Próbáld ki, mekkora a legkisebb kernel ami még működik. Sokat tanulhatsz abból, ha kihagyod a felesleges dolgokat!

Olvasd el a "The Linux Kernel"-t (URL alább), és közben keresd meg a forráskód azon részeit, amire hivatkozik. A könyv (amint ezt írom) a 2.0.33-as kernel verzióval foglalkozik, ami elég öreg már. Könnyebb követni, ha letöltöd ezt az öreg verziót, és ott olvasod a forrást. Fantasztikus érzés "process" és "page" nevű C kóddarabokat találni.

Hackelj! Próbálj meg extra üzeneteket vagy valami mást kiíratni.

4.3 További információ

- `/usr/src/linux/README` és a `/usr/src/linux/Documentation/` tartalma (Lehet, hogy ezek máshol vannak a Te rendszereden)
- *Kernel HOWTO* <http://mirror.aarnet.edu.au/linux/LDP/HOWTO/Kernel-HOWTO.html> (Kernel HOGYAN)
- A kernel beállításakor olvasható segítség a `make menuconfig` vagy `make xconfig` használatakor
- *The Linux Kernel (és más LDP útmutatók)* <http://mirror.aarnet.edu.au/linux/LDP/LDP/> (A Linux Kernel)
- forráskód, lásd a

  Building a Minimal Linux System from Source Code <http://www.netspace.net.au/~gok/power2bash> (Minimális Linux rendszer építése forrásból) doksit URL-ekért

5 A GNU C könyvtár

A következő dolog, ami a számítógép bekapcsolásakor történik, hogy betöltődik és lefut az init. Azonban az init - mint majdnem minden program - függvényeket használ más könyvtárakból.

Talán már láttál C nyelvű példaprogramot, mint ez is:

```
main() {
        printf("Hello World!\n");
}
```

A programban nincs leírva mi az a "`printf`", honnan is jön ez? Ez a standard C könyvtárakból - egy GNU/Linux rendszeren a glibc - származik. Ha Visual C++ alatt fordítod le, akkor ugyanannak a standard függvénynek a Microsoftos változatából jön. Milliárdnyi van ezekből a standard függvényekből a matematikához, stringekhez, dátumhoz/időhöz, memóriafoglaláshoz és így tovább. A Unix (beleértve a Linuxot) alatt minden vagy C-ben van írva, vagy nagyon úgy kell tennie mintha abban lenne írva. Egyszóval minden program tudja használni ezeket a függvényeket.

Ha belenézel a `/lib` könyvtárba, sok olyan fájlt fogsz látni, amit `libvalami.so`-nak vagy `libvalami.a`-nak hívnak. Ezek a függvények könyvtárai. A glibc csak a GNU változata ezeknek a függvényeknek.

Két módon használhatják a programok ezeket a függvénykönyvtárakat. Ha *statikusan* kapcsolódnak egy programhoz, ezek a függvények belemásolódnak a végrehajtható fájlba, amikor létrejön. Ez az, amire a `libvalami.a` könyvtárak valók. Ha *dinamikusan* kapcsolódnak egy programhoz (ez az alapértelmezés), akkor amikor a program fut és szüksége van a függvény kódjára, meghívja azt a `libvalami.so` fájlokból.

Az "`ldd`" parancsra van szükséged, ha meg akarod tudni, melyik függvénykönyvtárat használja egy bizonyos program. Itt vannak például azok a függvénykönyvtárak, amiket a "`bash`" használ:

```
[greg@Curry power2bash]$ ldd /bin/bash
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x40019000)
        libc.so.6 => /lib/libc.so.6 (0x4001d000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

5.1 Beállítás

Néhány a könyvtárakban lévő függvény attól függ, hogy helyileg hol vagy. Például mi itt Ausztráliában a dátumokat így írjuk: nn/hh/éé, de az amerikaiak hh/nn/éé -et írnak. Van egy program ami a "`glibc`" csomagban van, "`localedef`" a neve. Ez lehetővé teszi ezek beállítását.

5.2 Gyakorlatok

Használd az "`ldd`" parancsot, hogy megtudd, milyen programkönyvtárakat használnak a kedvenc alkalmazásaid.

Használd az "`ldd`"-t arra, hogy megtudd, milyen könyvtárakat használ az "`init`".

Készíts egy játék könyvtárat, egy vagy két függvénnyel. Az "`ar`" programmal lehet létrehozni őket. Az "`ar`" kézikönyv-oldalából megtudhatod ennek menetét. Írj, fordíts le és csatolj egy programot ami ezt a könyvtárat használja.

5.3 További információ

- forráskód, lásd a

  Building a Minimal Linux System from Source Code <http://www.netspace.net.au/~gok/power2bash>

  doksit URL-ért

6 Init

Csak a "System V" stílusú "init"-ről beszélek, amit a Linux rendszerek a leginkább használnak. Vannak más megoldások is. Valójában bármilyen programot berakhatsz az `/sbin/init`-be, amit a kernel le fog futtatni, amint betöltődött.

Az "`init`" feladata, hogy minden úgy működjön ahogy kell. Leellenőrzi a fájlrendszereket és felcsatolja őket. Lefuttatja a "démonokat", hogy naplózzák a rendszerüzeneteket, hálózatkezelést csináljanak, kiszolgáljanak web oldalakat, figyeljenek az egeredre stb. Szintén ez indítja a getty folyamatokat, amik kirakják a bejelentkező (login) promptot a virtuális terminálodra.

Van egy nagyon komplikált történet a "futási szintek" (run-levels) közötti kapcsolásról, de a legnagyobb részét kihagyom, és csak a rendszer indulásáról beszélek.

Az" init" beolvassa az `/etc/inittab` fájlt, ez írja le mit csináljon. Általában először lefuttat egy inicializáló szkriptet. A program, ami végrehajtja (soronként) ezt a szkriptet, a bash, ugyanaz a program ami kiírja a parancssorodat. Debian rendszereken az inicializáló szkript az `/etc/init.d/rcS`, Red Hat-en a `/etc/rc.d/rc.sysinit`. Ez az, ami a fájlrendszereket leellenőrzi és felcsatolja, beállítja az órát és a csere (swap) területet, beállítja a gép hálózati nevét (hostname) stb.

Ezután egy másik szkript indul el, ami elvisz az alapértelmezett futási szintre. Ez csak annyit jelent, hogy elindul néhány alrendszer. Van egy pár könyvtár, az `/etc/rc.d/rc0.d, /etc/rc.d/rc1.d, ...,`

`/etc/rc.d/rc6.d` Red Hat alatt, vagy az `/etc/rc0.d , /etc/rc1.d, ..., /etc/rc6.d` Debian alatt, ami a futási szintekhez tartozik. Ha a 3-as futási szinten vagyunk Debianban, akkor a szkript lefuttatja az összes "S"-el (start rövidítése) kezdődő szkriptet az `/etc/rc3.d`-ben. Ezek a szkriptek valójában csak linkek más szkriptekhez egy másik könyvtárban, amit általában "`init.d`"-nek hívnak.

Tehát az "`init`" meghívja a futási szint (run-level) szkriptünket, és más "S"-el kezdődő szkripteket keres egy könyvtárban. Megtalálhatja elsőként az `S10syslog`-ot. A számok megmondják a (run-level) szkriptnek, milyen sorrendben futtassa le őket. Ebben az esetben az `S10syslog` hajtódik végre először, mivel nincs S00 ... S09 kezdetű szkript. De az `S10syslog` valójában csak egy link az `/etc/init.d/syslog`-hoz, ami egy rendszernaplózást (system logger) elindító és leállító szkript. Mivel a link "S"-el kezdődik, a (run-level) szkript tudja, hogy végre kell hajtania a `syslog`-ot egy "start" paraméterrel. Vannak ennek megfelelő, "K"-val (kill - megölni) kezdődő linkek, amik meghatározzák, mi és milyen sorendben legyen leállítva, amikor elhagyjuk a futási szintet.

Ha meg akarod változtatni az alrendszerek alapértelmezett indítását, létre kell hoznod ezeket a linkeket az `rcN.d` könyvtárban, ahol N az alapértelmezett futási szint az `inittab` fájlodban.

Az utolsó fontos dolog, amit az "init" csinál, hogy elindít pár `getty`-t. Ezek "respawned" állapotúak, ami azt jelenti, hogy ha leállnak, akkor az "`init`" egyszerűen újraindítja őket. A legtöbb disztribució hat virtuális terminált indít. Lehet, hogy kevesebbet akarsz a memóspórolás érdekében, vagy többet, amivel egy csomó dolgot hagyhatsz futni és gyorsan váltogathatsz köztük ha szükséged van rájuk. Lehet, hogy akarsz majd futtatni egy `getty`-t szöveges terminálnak vagy modemes betárcsázáshoz. Ebben az estben át kell szerkesztened az `inittab` fájlodat.

6.1 Beállítás

A `/etc/inittab` az "init" legfelső szintű konfigurációs fájlja.

Az `rcN.d` könyvtárak, ahol N = 0, 1, ..., 6, határozzák meg, milyen alrendszerek indulnak el.

Valahol az "init" által elindított szkriptekben, a `mount -a` parancs kerül végrehajtásra. Ez annyit jelent, hogy felcsatolódik az összes fájlrendszer, aminek fel kell csatolódnia. Ez az `/etc/fstab` fájlban van leírva. Ha meg akarod változtatni, hogy hova és mi csatolódjon, amikor a rendszered elindul, ezt a fájlt kell átszerkesztened. Az `fstab` fájlnak van egy kézikönyv (man) oldala is.

6.2 Gyakorlatok

Keresd meg alapértelmezett futási szintedhez tartozó `rcN.d` könyvtáradat és adj ki egy `ls -l` parancsot, hogy lásd milyen fájlokhoz kapcsolódnak a linkek.

Változtasd meg a rendszereden futó getty-k számát.

Törölj minden alrendszert, amire nincs szükséged, az alapértelmezett futási szinten.

Nézd meg, milyen közel jutottál a kezdéshez.

Telepíts egy floppy-ra lilo-t, kernelt és egy statikusan linkelt "hello world" programot, amit nevezz el `/sbin/init`-nek, és figyeld ahogy bebootol és kiírja: hello.

Nézd figyelmesen a rendszered indulását és jegyzetelj, milyen üzeneteket ír ki arról ami történik. Esetleg nyomtass ki egy részt a rendszernaplóból (`/var/log/messages`). Ezután az `inittab`-bal kezdve, fusd át az összes szkriptet és figyeld meg, melyik kódrészlet mit csinál. Berakhatsz extra start üzeneteket is, mint például:

```
echo "Hello, en vagyok a rc.sysinit"
```

Ez jó gyakorlat arra is, hogy megtanuld a Bash shell alatti szkriptírást. A szkriptek közül néhány eléggé komplikált. Legyen kéznél egy jó Bash referenciakönyv.

6.3 További információ

- Létezik kézikönyv-oldal az `inittab` és `fstab` fájlokhoz. Megnézheted a "`man inittab`" begépelésével.
- A Linux System Administrators Guide tartalmaz egy

  jó részt <http://mirror.aarnet.edu.au/linux/LDP/LDP/> az init-ről.
- forráskód, lásd a

  Building a Minimal Linux System from Source Code <http://www.netspace.net.au/~gok/power2bash> doksit URL-ért.

7 A fájlrendszer

Ebben a fejezetben a fájlrendszer szót két különböző értelemben fogom használni. Vannak fájlrendszerek a lemez particiókon és más eszközökön, és van egy fájlrendszer, amit a futó Linux alól látsz. A Linuxban "felcsatolod" (mount) a lemezes fájlrendszereket az operációs rendszer fájlrendszerére.

Az előző fejezetben megemlítettem, hogy az" init" szkriptek leellenőrzik és felcsatolják a fájlrendszereket. Ezt az "`fsck`" és a "`mount`" parancsok végzik el.

A merevlemez csak egy nagy tárhely, ahova egyeseket és nullákat írhatsz. A fájlrendszer erre rárak egy szerkezetet, és ettől néz ki úgy, mintha fájlok lennének könyvtárakon belül és azok is további könyvtárakon belül... Minden fájlnak van egy leíró táblája (inode), ami nyilvántartja kié a fájl, mikor hozták létre és hol vannak a részei. A könyvtárakat is inode-ok írják le, de ezekben a tárolódik, hogy hol találhatók azoknak a fájloknak az inode-jai, amik a könyvtárban vannak. Ha a rendszer olvasni akarja a `/home/greg/bigboobs.jpeg` fájlt, először megkeresi a gyökér (root) / könyvtár leíró táblját a "superblock"-ban, ezután megkeresi a / tartalmában lévő `home` könyvtár inode-ját, ezután megkeresi a `/home` tartalmában lévő `greg` könyvtár inode-ját, ezután megkeresi a `bigboobs.jpeg` leíró táblját, ami jelzi melyik lemezblokkot kell olvasnia.

Ha hozzáfűzünk egy kis adatot egy fájl végéhez, megeshet, hogy először az adat íródik ki, mielőtt a leíró tábla frissülne (ami jelezné, hogy az új blokkok a fájlhoz tartoznak), vagy fordítva. Ha ezen a ponton áramszünet következik be, megsérülhet a fájlrendszer. Ez az a dolog, amit az "`fsck`" megpróbál felderíteni és kijavítani.

A "mount" parancs egy eszközön lévő fájlrendszert hozzáad a használatban lévő rendszered hierarchiájához. Általában a kernel csak olvasható módban csatolja fel a (root) fájl rendszert. A "mount" parancsot használják arra, hogy újra felcsatolják írható-olvasható módban, miután az "`fsck`" leellenőrizte, hogy minden rendben.

A Linux támogat másfajta fájlrendszereket is: msdos, vfat, minix stb. Egy bizonyos fájlrendszer részletei el vannak vonatkoztatva egy virtuális fájlrendszerre (VFS). Nem megyek bele a részletekbe, van egy vita erről a "A Linux Kernel"-ben (lásd a 4.3 (A Linux Kernel fejezetet) URL-ért).

Egy teljesen más fájlrendszer csatolódik be a `/proc` könyvtárba. Valójában ez csak a kernelben lévő dolgok megjelenítése. Mindegyik - a rendszerben futó folyamatnak (process) - van egy könyvtára, ahol a folyamat száma a könyvtár neve. Vannak ott fájlok is, mint például `interrupts` (megszakítások) és `meminfo`, amik elárulják, hogy hogyan használja a gép a hardvert. Nagyon sokat tanulhatsz a `/proc` felfedezése közben.

7.1 Beállítás

Vannak az "`mke2fs`" parancsnak (ami ext2 fájlrendszert hozza létre) paraméterei is. Ezek ellenőrzik a blokkok méretét, a leíró táblák (inode) számát stb. Olvasd el az "`mke2fs`" kézikönyv-oldalát részletekért.

Hogy mi hova csatolódik a fájlrendszereden, az `/etc/fstab` fájlban dől el. Ennek is van kézikönyv-oldala.

7.2 Gyakorlatok

Készíts egy nagyon kicsi fájlrendszert, és nézd meg egy hexa editorral. Azonosítsd az inode-okat, szuperblokkokat és a fájl összetevőket.

Biztos vagyok benne, vannak eszközök amik visszadják a fájlrendszered grafikus kinézetét. Keress egyet, próbáld ki, és küldj egy URL-t és leírást nekem e-mailben!

Nézd meg az ext2 fájlrendszer kódját a kernelben.

7.3 További információ

- A "The Linux Kernel" című LDP könyv 9. fejezete kitűnő leírást ad a fájlrendszerekről. Megtalálhatod például. a TLDP egyik ausztrál

  tükröoldalán <http://mirror.aarnet.edu.au/linux/LDP/LDP/> .

- A "`mount`" parancs az util-linux csomag része, van egy hivatkozás hozzá a

  Building a Minimal Linux System from Source Code <http://www.netspace.net.au/~gok/power2bash> doksiban.

- A "`mount`, "`fstab`", "`fsck`"," `mke2fs`" és a `proc` kézikönyv-oldalai.

- A `Documentation/proc.txt` fájl a Linux forrásában elmagyarázza a `/proc` fájlrendszert.

- Az EXT2 fájlrendszer segédprogramjai, az

  ext2fsprogs <http://web.mit.edu/tytso/www/linux/e2fsprogs.html> honlapja, vagy az

  ext2fsprogs <ftp://mirror.aarnet.edu.au/pub/linux/metalab/system/filesystems/ext2/> ausztrál tükröoldal. Itt van még egy Ext2fs-áttekintés, bár már régi és nem olyan olvasmányos, mint a "The Linux Kernel" 9. fejezete.

- *Unix Fájlrendszer Standard* <ftp://tsx-11.mit.edu/pub/linux/docs/linux-standards/fsstnd/> . Másik *link* <http://www.pathname.com/fhs/> a Unix Fájlrendszer Standard-hoz. Ez leírja, minek hova kell kerülnie egy Unix fájlrendszerben, és miért. Szintén ebben található egy leírás arról, hogy a `/bin`, `/sbin` stb. könyvtáraknak mit kell tartalmaznia. Jó referencia, ha a célod egy minimális, mégis teljes rendszer megépítése.

8 Kernel démonok

Ha kiadod a "`ps aux`" parancsot, valami ehhez hasonlót fogsz látni:

```
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
root         1  0.1  8.0  1284   536   ? S    07:37   0:04 init [2]
root         2  0.0  0.0     0     0   ? SW   07:37   0:00 (kflushd)
root         3  0.0  0.0     0     0   ? SW   07:37   0:00 (kupdate)
root         4  0.0  0.0     0     0   ? SW   07:37   0:00 (kpiod)
```

```
root         5  0.0  0.0     0     0   ? SW   07:37   0:00 (kswapd)
root        52  0.0 10.7  1552   716   ? S    07:38   0:01 syslogd -m 0
root        54  0.0  7.1  1276   480   ? S    07:38   0:00 klogd
root        56  0.3 17.3  2232  1156   1 S    07:38   0:13 -bash
root        57  0.0  7.1  1272   480   2 S    07:38   0:01 /sbin/agetty 38400 tt
root        64  0.1  7.2  1272   484  S1 S    08:16   0:01 /sbin/agetty -L ttyS1
root        70  0.0 10.6  1472   708   1 R   Sep 11   0:01 ps aux
```

Ez a rendszerben futó folyamatok listája. Az információ a `/proc` fájlrendszerből (amit az előző részben már említettem) származik. Figyeld meg, hogy az "`init`" az egyes számú folyamat. A "kflushd", "kupdate", "kpiod" és a "kswapd" a 2, 3, 4 és 5-ös számú folyamatok. Van azonban valami furcsa: nézd meg mind a virtuális tárolási méret (SIZE), mind a valódi tárolási méret (RSS) nevű oszlopokat. Ezeknél a folyamatoknál nullák szerepelnek. Hogyhogy egy folyamat nem használ memóriát?

Ezek a folyamatok a kernel démonok. A kernel legnagyobb része egyáltalán nem mutatkozik a folyamatlistákon. Csak úgy találhatod ki mennyi memóriát használnak, ha kivonod a gépben lévő memóriából a rendelkezésre álló memória mennyiségét. A kernel démonok az "init" után indulnak el, így kapnak folyamatszámot ugyanúgy, mint a normál folyamatok. De a kódjuk és adataik a kernel részére lefoglalt memóriában van.

A COMMAND oszlop bejegyzéseinél zárójelek vannak, mivel a `/proc` fájlrendszer nem tartalmaz parancssori információkat ezekről a folyamatokról.

Tehát mire is valók ezek a kernel démonok? E dokumentum előző verzióiban volt egy segítségkérő rész, mivel nem tudtam sokat róluk. A következő történet az erre adott válaszokból (amiért nagyon hálás vagyok) lett összerakva. Szívesen fogadom a további okosításokat, referenciákat és kiigazításokat!

A bevitel és kivitel *puffereken* (buffers) keresztül történik a memóriában. Ettől gyorsabban mennek a dolgok. Amit a programok írnak, a memóriában lehet tartan - egy pufferben - ezután nagyobb (hatékonyabb) darabokban lehet a lemezre írni. A "`kflushd`" és "`kupdate`" démonok ezt a munkát végzik: a "`kupdate`" periodikusan lefut (5 másodpercenként?), hogy leellenőrizze van-e használt puffer (dirty buffers). Ha igen, utasítja a "`kflushd`" démont, hogy írja ki őket a lemezre.

A folyamatoknak gyakran semmi dolguk, és azoknak amik éppen futnak, gyakran nincs arra szükségük, hogy az össszes kódjukat és adataikat a memóriában tartsák. Ez azt jelenti, hogy jobb használt vesszük a memóriánknak, ha a futó programok éppen nem használt részeit kiírjuk a merevlemez csere (swap) partíciójára. Az adat szükséges ki- és bemozgatását a memóriába (-ból) a "`kpiod`" és a "`kswapd`" végzi. Minden másodpercben vagy valahogy így, a "`kswapd`" felébred, hogy leellenőrizze a memória helyzetét, és ha szükség van valamire a memóriában - ami a lemezen található - vagy nincs elég szabad memória, a "`kpiod`" hívódik meg.

Lehet, hogy a "`kapmd`" démon is fut a rendszereden, ha az automatikus áramgazdálkodás (automatic power management) be van fordítva a kerneledbe.

8.1 Beállítás

Az "`update`" programal lehet beállítani a "`kflushd`" és a "`kswapd`" démonokatt. Próbáld az "`update -h`" parancsot kiadni valami információért.

A csere partíciót be lehet kapcsolni a "`swapon`", és ki lehet kapcsolni a "`swapoff`" paranccsal. Az init szkript (`/etc/rc.sysinit` vagy `/etc/rc.d/rc.sysinit`) általában meghívja a "`swapon`"-t, amint a rendszer elindul. Azt mondták nekem, hogy a "`swapoff`" praktikus, ha áramot akarunk spórolni laptopon.

8.2 Gyakorlatok

Csinálj egy `update -d`-t, és figyeld a dumát az utolsó sorban a "threshold for buffer fratricide"-ről. Most már van egy érdekes fogalmad, járj utána!

Lépj be a `/proc/sys/vm` könyvtárba és adj ki egy "`cat`" parancsot az ott lévő fájlokra. Figyeld meg, mi mindenre tudsz rájönni ezáltal.

8.3 További információ

A Linux Dokumentációs Projekt "The Linux Kernel" dokumentuma (lásd 4.3 (A Linux kernel) fejezetet URL-ért)

A Linux kernel forráskódja, ha elég bátor vagy! A "`kswapd`" kódja a `linux/mm/vmscan.c`-ben, a "`kflushd`" és "`kupdate`" kódja pedig a `linux/fs/buffer.c` fájlban van.

9 Rendszernaplózó

Az "init" elindítja a "`syslogd`" és a "`klogd`" démonokat. Ezek üzeneteket írnak a naplófájlokba. A kernel üzeneteit a "`klogd`" kezeli, míg a "`syslogd`" más folyamatok üzeneteit intézi. A fő naplófájl a `/var/log/messages`. Jól teszed, ha először itt keresel ha valami baj van a rendszereddel. Gyakran található ott egy fontos nyom.

9.1 Beállítás

Az `/etc/syslog.conf` fájl írja le a naplózóknak, hogy milyen üzenetet hova írjanak. Az üzeneteketet az azonosítja, hogy milyen folyamattól és milyen prioritással érkeznek. Ez a fájl olyan sorokból áll, amik jelzik, hogy az x folyamattól y prioritással érkező üzenetek z-hez menjenek, ahol z lehet egy fájl, tty, nyomtató, távoli ügyfél (remote host) vagy bármi más.

FIGYELEM: A Syslognak szüksége van az `/etc/services` fájlra. A services fájl portokat jelöl ki. Nem vagyok benne biztos, hogy a syslognak kell egy port, amivel így tud naplófájlt készíteni távoli gépre is (remote logging), vagy ahhoz is egy portra van szükség, hogy a helyi gépre naplófájlt készítsen (local logging), vagy csak az `/etc/services`-t használja, hogy az általad beírt folyamatneveket az `/etc/syslog.conf` port számokká konvertálja.

9.2 Gyakorlatok

Vess egy pillantást a rendszernaplódra. Keress egy üzenetet amit nem értesz, és találd ki mit jelent.

Küld ki az üzeneteidet egy tty-re. (állítsd vissza miután megcsináltad)

9.3 További információ

Ausztrál sysklogd *tükör* <http://mirror.aarnet.edu.au/pub/linux/metalab/system/daemons/>

10 Getty és bejelentkezés

A getty program teszi lehetővé, hogy bejelentkezzhess egy soros eszközön keresztül mint például egy virtuális terminál, szöveges terminál, vagy egy modem. Kiírja a bejelentkező parancssort (login prompt). Miután

beírtad a felhasználói nevedet, a getty átadja azt a "`login`"-nak, ami megkérdezi a jelszót, leellenőrzi, és ad egy shellt.

Sok használható getty van. Néhány disztribució- beleértve a Red Hat-et - egy nagyon kicsit használnak, aminek a neve "`mingetty`", ez csak virtuális terminálokkal működik.

A "`login`" program az util-linux csomag része, amiben található egy másik getty is, aminek "`agetty`" a neve, és jól működik. Szintén ebben a csomagban van az `"mkswap", "fdisk", "passwd", "kill", "setterm", "mount", "swapon", "rdev", "renice", "more"` és még sok más program.

10.1 Beállítás

Az üzenet, amit a képernyőd tetején látsz a login prompt-al együtt, az `/etc/issue` fájlból származik. A getty-kat általában a `/etc/inittab`-ból indítjuk. A login a felhasználói jellemzőket a `/etc/passwd`-ben ellenőrzi, illetve, ha árnyékjelszavakat használsz, a `/etc/shadow`-ban.

10.2 Gyakorlatok

Készíts "kézzel" egy `/etc/passwd` fájlt. A jelszavak üresek lehetnek, és a "`passwd`" programmal változtathatod meg, ha belépsz. Lásd a fájl kézikönyv- oldalát. Használd a "`man 5 passwd`" parancsot ehhez, hogy ne a program, hanem a fájl kézikönyvét kapd meg.

11 Bash

Ha adsz a "`login`"-nak egy érvényes felhasználói név - jelszó párost, ellenőrzi azt az `/etc/passwd` állományban, hogy megtudja melyik shell-t indítsa el neked. Linux rendszeren ez legtöbbször a "`bash`". A "`bash`" dolga a parancsaid értelmezése, illetve gondoskodni ezek végrehajtásáról. Egyszerre felhasználói felület és programnyelv-értelmező (programming language interpreter - a lektor).

Mint felhasználói felület beolvassa parancsaidat, és végre is hajtja ezeket - amennyiben un. "belső" parancsok, mint például a "`cd`" - illetve megkeresi és futtatja a programot, - ha az egy un. "külső" parancs, mint például a "`cp`" vagy a "`startx`". Olyan megszokott dolgokat is csinál, mint például megjegyzi az előzőleg kiadott parancsokat valamint kiegészíti a fájlneveket.

Már láttuk a "`bash`"-t működni, mint programnyelv-értelmező. A szkriptek, amiket az "`init`" a rendszer elindításáért futtat, általában shell szkriptek, amelyeket a "`bash`" hajt végre. Egy rendes programnyelv, a szokásos parancssoros rendszereszközökkel nagyon hatásos kombinációt alkot, ha tudod mit csinálsz. Például sokat kellett foltoznom (patch - a ford.) egy könyvtárnyi forráskódon a minap. Mindezt egyetlen sorral végre tudtam hajtani:

```
for f in /home/greg/sh-utils-1.16*.patch; do patch -p0 < $f; done;
```

Ez megvizsgálja az összes olyan fájlt a home könyvtáramban, aminek a neve "`sh-utils-1.16`"-al kezdődik, és "`.patch`"-el végződik. Mindezt ciklusban teszi, beállítja az "`f`" változót, majd végrehajtja a "`do`" és "`done`" közötti parancsot. Ebben az esetben 11 folt volt, de könnyen lehetett volna akár 3000 is.

11.1 Beállítás

Az `/etc/profile` fájl határozza meg a bash viselkedését/működését a teljes rendszerre vonatkozóan. Amit ide beraksz, az hatással lesz mindenkire, aki a bash shellt használja a rendszereden. Különböző dolgokat fog csinálni, például könyvtárakat ad hozzá a "`PATH`"-hoz, vagy beállítja a "`MAIL`" környezeti változót.

A billentyűzet alapbeállítása gyakran kívánnivalót hagy maga után. A readline az, ami ezt irányítja. Ez egy különálló csomag, ami a parancssori csatolófelületet irányítja, rendelkezésre bocsátja a parancssori történetet (command history) és fájlnév-kiegészítést, csakúgy mint néhány fejlettebb sorszerkesztési módot. Ez bele van szerkesztve a bash-ba. Alapértelmezésként a readline a home könyvtáradban lévű `.inputrc` fájl használatára van beállítva. Az INPUTRC változó használható arra, hogy a bash számára felülírjuk ezt. Például a Red Hat 6 rendszerben az "`INPUTRC`" értéke `/etc/inputrc` az `/etc/profile` fájlban. Ez azt jelenti, hogy a backspace, delete, home és end billentyűk mindenkinek nagyszerűen működnek.

A bash a teljes rendszerre vonatkozó beállítások beolvasása után a személyes beállításokat tartalmazó fájlt keresi meg. Ellenőrzi a home könyvtáradat, `.bash_profile`, `.bash_login` és `.profile` fájlok után kutatva. Ezek közül az első létezőt futtatja. Ha a bash működésén szeretnél változtatni úgy, hogy a többiekre ne legyen hatással, akkor azt itt tedd meg. Például sok alkalmazás környezeti változókat használ a működésének beállítására. Nekem van egy "`EDITOR`" változóm, ami a "`vi`" programra van beállítva, így én ezt használhatom a Midnight Commanderben (egy kiváló karakteres fájlkezelő) az ő saját szövegszerkesztője helyett.

11.2 Gyakorlatok

A bash alapjait könnyű megtanulni. De ne állj meg itt: hihetetlen mélységei vannak. Tedd szokásoddá, hogy mindig jobb utat keresel egy dolog véghezvitelére.

Olvass shell szkripteket, keresd meg azokat a részeket, amelyeket nem értesz.

11.3 További információ

- Létezik egy "Bash Reference Manual" (Bash referencia kézikönyv) amely részletes, de nehezen érthető
- Van egy O'Rielly könyv amely a Bash-ról szól, de nem vagyok biztos abban, hogy ez jó
- én nem ismerek több szabadon használható, jó és naprakész ismertetőt a "bash"-ról. Ha Te igen, kérlek küld el nekem e-mailben az URL-t
- forráskódokra mutató hivatkozásokért látogass el a

  Building a Minimal Linux System from Source Code <http://www.netspace.net.au/~gok/power2bash>

  honlapra

12 Parancsok

A legtöbb dolgot a bash-ban olyan parancsok segítségével végezheted el, mint a `cp`. Ezen parancsok legtöbbje kis program, bár van néhány - például a "`cd`" - amely be van építve a shell-be.

A parancsok csomagokban vannak, legtöbbjüket a Free Software Foundation (Szabad Szoftver Alapítvány) készíti. Ahelyett, hogy itt felsorolnám a csomagokat, átirányítalak a

Linux From Scratch HOWTO <http://www.linuxfromscratch.org> (Linux a kezdetektől HOGYAN) honlapra. Ez egy teljes és frissen tartott listája a Linux rendszerben lévő csomagoknak, valamint útmutatás ahhoz, hogyan készítsük el ezeket.

13 Befejezés

Az egyik legjobb dolog a Linuxban - szerény véleményem szerint - hogy a belsejébe nézhetsz és megtudhatod miként működik. Remélem, legalább annyira élvezed ezt mint én, és ez kis jegyzet segít abban, hogy ezt megtedd.

14 Adminisztráció

14.1 Szerzői jog

14.2 Magyar fordítás

A magyar fordítást *Koller Csaba* <mailto:ckoller@mailbox.hu_NO_SPAM> készítette (2002.11.25). A lektorálást *Daczi László* <mailto:dacas@freemail.hu_NO_SPAM> és *Szíjjártó László* <mailto:laca@janus.gimsz.sulinet.hu_NO_SPAM> végezte el (2003.07.18). Bármilyen fordítással kapcsolatos észrevételt a *linuxhowto@sch.bme.hu* <mailto:linuxhowto@sch.bme.hu_NO_SPAM> címre küldjetek. Eme dokumentum legfrissebb változata megtalálható a *Magyar Linux Dokumentációs Projekt* <http://tldp.fsf.hu/index.html> honlapján.

14.3 Honlap

A dokumentum legutolsó változatát a

From Powerup To Bash Prompt <http://www.netspace.net.au/~gok/power2bash> honlapon találod meg, csakúgy mint a párját, a "Building a Minimal Linux System from Source Code" leírást.

A francia fordítás a

From Powerup To Bash Prompt <http://www.freenix.fr/unix/linux/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html> honlapon található, köszönet érte Dominique van den Broeck-nek. A japán fordítást Yuji Senda készíti el hamarosan, ha nem lenne a

Japanese Documentation and FAQ Project <http://www.linux.or.jp/JF>

honlapon máris.

14.4 Visszajelzés

Bármilyen megjegyzésedről, kritikádról és ajánlásodról szeretnék hallani, a dokumentum fejlesztése érdekében. Kérlek küld el ezeket nekem:

Greg O'Keefe <mailto:gcokeefe@postoffice.utas.edu.au> .

14.5 Köszönetnyilvánítások

A terméknevek az illető cégek által bejegyzett védjegyek, erre való tekintettel ez itt megemlítésre került.

Szeretnék pár embernek köszönetet mondani azért, hogy segítettek ennek a doksinak a létrehozásában.

Michael Emery

Emlékeztetett az Unios-ra.

Tim Little

Néhány ötletet adott az `/etc/passwd`-höz.

sPaKr on #linux in efnet

Rámutatott, hogy a syslogd-nak az `/etc/services`-re van szüksége, és megismertetett a "rolling your own" kifejezéssel annak érdekében, hogy leírjam egy rendszer forráskódból történő felépítésének menetét.

Alex Aitkin

Felhívta a figyelmem Vico-ra, az ő mondása a "verum ipsum factum" (jelentése: a dolgokat készítésük közben ismerjük meg igazán).

Dennis Scott

Kijavította a hexadecimális számításaimat.

jdd

Rámutatott néhány helyesírási hibára.

David Leadbeater

Közreműködött a kernel démonokról szóló rész megalkotásában.

Dominique van den Broeck

Lefordította a dokumentumot franciára.

Matthieu Peeters

Néhány hasznos információt adott a kernel démonokról.

John Fremlin

Néhány hasznos információt adott a kernel démonokról.

Yuji Senda

Elkészítette a japán fordítást.

Antonius de Rozari

Közreműködött a GNU assembler UNIOS változatának létrehozásában (lásd a források fejezetet a honlapon).

14.6 Változások története

14.6.1 0.8 -> 0.9 (2000 november)

- Hozzáadtam néhány Matthieu Peeterstől és John Fremlin-től származó információt a kernel démonokról és a `/proc` fájlrendszerről.

14.6.2 0.7 -> 0.8 (2000 szeptember)

- Kivettem a rendszerépítésről szóló útmutatót, és egy külön dokumentumba raktam, ezért néhány hivatkozást beszúrtam.
- A honlapot átraktam a *learning@TasLUG* <http://learning.taslug.org.au/power2bash> -ról, *a saját tárhelyemre* <http://www.netspace.net.au/~gok/power2bash> .
- Nem sikerült beraknom a dokumentumba sok - különböző emberektől származó - hasznos anyagot. Talán legközelebb :(

14.6.3 0.6 -> 0.7

- nagyobb hangsúly a magyarázaton és kissebb a rendszerépítésen. A rendszerépítési információt külön fejezetbe raktam és megrövidítettem. Az ez iránt érdeklődők olvassák el Gerard Beekman "Linux From Scratch" (Linux a kezdetektől) című művét.
- David Leadbeater néhány magyarázatának hozzáadása
- pár hivatkozás javítása, és egy hozzáadása az unios letöltéséhez a learning.taslug.org.au/resources webhelyről
- hivatkozások tesztelése és javítása
- általános újraírás és csinosítás

14.6.4 0.5 -> 0.6

- a változások történetének hozzáadása
- néhány teendő hozzáadása

14.7 Tennivalók

- elmagyarázni mindent a következőkről: kernel modulok, depmod, modprobe, insmod (először utána kell néznem!)
- megemlíteni a /proc fájlrendszert itt
- docbook sgml formátumba konvertálni a doksit
- további útmutatók hozzáadása - talán a nagyobb útmutatásokból külön fejezetet készítve - mint például egy minimális rendszer készítése egy linux összeállításból, fájlról-fájlra
- a makefile hack-et hozzáadni a bash építési útmutatóhoz - lásd az "easter" megjegyzésekben.