Gyakran Ismételt Kérdések a FreeBSD 6.X, 7.X és 8.X változatairól

Kivonat

Ezek a gyakran ismételt kérdések a FreeBSD 6.*X*, 7.*X* és 8.*X* változataira vonatkoznak. Az összes bejegyzés a FreeBSD 6.*X* vagy annál újabb változataira vonatkozik, hacsak azt külön nem jelezzük. Ha szeretnénk segíteni a projektnek, akkor küldjünk egy levelet a FreeBSD Dokumentációs Projekt levelezési lista címére! Ennek a dokumentumnak a legfrissebb változata mindig elérhetõ a FreeBSD World Wide Web szerverérõl. HTTP-n keresztül letölthetõ egyetlen nagy HTML állományként, vagy a FreeBSD FTP szerverérõl szöveges, PostScript® PDF stb. formátumban. Továbbá keresni is tudunk a GYIK-ban.

Fordította: Páli Gábor, utolsó ellenõrzés: 2010.11.28.

Tartalomjegyzék

1. Bevezetés . . . 3
2. Dokumentációs és támogatás . . . 9
3. Telepítés . . . 13
4. Hardverkompatibilitás . . . 24
5. Hibaelhárítás . . . 34
6. Kereskedelmi alkalmazások . . . 50
7. Felhasználói alkalmazások . . . 52
8. A rendszermag beállítása . . . 57
9. Lemezek, állományrendszerek és rendszertöltők . . . 60
10. Rendszeradminisztráció . . . 74
11. Az X Window System és a virtuális konzolok használata . . . 85
12. Hálózatok . . . 95
13. Biztonság . . . 103
14. PPP . . . 107
15. Soros vonali kommunikáció . . . 122
16. Egyéb kérdések . . . 127
17. Mókás dolgok a FreeBSD-vel kapcsolatban . . . 132
18. Témák haladóknak . . . 136
19. Köszönetnyilvánítás . . . 144
Irodalomjegyzék . . . 145

Chapter 1. Bevezetés

Üdvözöljük a FreeBSD 6.*X*-8.*X* Gyakran Ismételt Kérdéseiben!

Hasonlóan a Usenetes GYIK-okhoz, ennek a dokumentumnak is az a célja, hogy a FreeBSD operációs rendszerrel kapcsolatban feltegye a legyakrabban ismételt kérdéseket (és persze megválaszolja ezeket!). Habár eredetileg azért íródott, hogy megspórolja a feleslegesen elvesztegetett sávszélességet és hogy megelõzze a régóta ismert kérdések újbóli feltételét, a GYIK idõközben egy értékes információforrássá is vált.

Igyekeztünk minden megtenni annak érdekében, hogy a GYIK a lehetõ legtöbb információt szolgáltassa. Ha szeretnénk javaslatokat tenni a továbbfejlesztésére, írjunk bátran a FreeBSD Dokumentációs Projekt levelezési lista címére!

1.1. Mi az a FreeBSD?

Ha tömören akarjuk összefoglalni, akkor a FreeBSD egy AMD64, Intel® EM64T, i386™, PC-98, IA-64, ARM®, PowerPC® és UltraSPARC® platformokra fejlesztett UNIX®-szerû operációs rendszer, amely a Kaliforniai Egyetem (Berkeley) rendszerének "4.4BSD-Lite" kiadására épül, valamint a "4.4BSD-Lite2" kiadásból tartalmaz még néhány továbbfejlesztést. Továbbá közvetett módon még felhasználja a Berkeley "Net/2" kiadásának i386™ architektúrára készített portját, a "386BSD" forrásait is, amit annak idején William Jolitz készített, noha ebbõl ténylegesen már csak nagyon kevés található a rendszerben. A FreeBSD részletesebb bemutatása és annak tulajdonságai a FreeBSD honlapján találhatóak.

A FreeBSD-t munkához, oktatáshoz és szórakozáshoz rengeteg cég, internetszolgáltató, kutató, informatikus, diák és otthoni felhasználó használja a világ minden táján.

A FreeBSD bõvebb bemutatásához olvassuk el a FreeBSD kézikönyvet.

1.2. Mi a FreeBSD Projekt célja?

A FreeBSD Projektnek az a célja, hogy olyan szoftvereket állítson elõ, amelyek tetszõlegesen felhasználhatóak, mindenféle kötöttségek nélkül. A fejlesztõk közül sokan nagyon sok idõt és munkát fektetnek a forráskódba (és így a Projektbe), ami nyilván megérdemelne némi anyagi ellensúlyozást olykor, de egyáltalán nem ragaszkodunk hozzá. Úgy érezzük, mindenek elõtt az a "küldetésünk", hogy feltétel nélkül segítsünk mindenkit a munkánkkal, függetlenül annak szándékaitól, így a munkánk a lehetõ legnagyobb körben kerül felhasználására és így nyújtja a lehetõ legtöbb hasznot. Véleményünk szerint ez az egyik legalapvetõbb célja a szabad szoftvereknek és ezt a hozzáállást támogatjuk a leginkább.

A forrásaink között található, GNU General Public License (GPL) vagy a GNU Library General Public License (LGPL) licencelésû munkák azonban már valamivel több kötöttséggel járnak, habár ezek inkább a közzétételükre vonatkoznak, nem pedig annak ellenkezõjére, ahogy azt általában megszokhattuk. A GPL licencû szoftverek kereskedelmi célú felhasználásának további esetleges nehézségei miatt azonban lehetõségeink szerint igyekszünk ezeket olyan szoftverekkel felváltani, amelyek a kevésbé szigorúbb FreeBSD licencet alkalmazzák.

1.3. A FreeBSD licenc tartalmaz valamilyen megszorítást?

Igen. Ezek a megszorítások azonban nem az adott munka felhasználását szabályozzák, hanem csupán azt, hogy miként viszonyuljunk a FreeBSD Projekthez. Ha komoly kétségeink lennének a licenceléssel kapcsolatban, olvassuk a jelenleg érvényes licencet (angolul). Az egyszerû kíváncsiskodók kedvéért nagyjából így tudnánk összefoglalni a licencet:

- Ne állítsuk, hogy mi készítettük.
- Ne pereljük a Projektet, ha nem mûködik.

1.4. A FreeBSD képes kiváltani a jelenleg használt operációs rendszerünket?

A legtöbb ember számára igen. A kérdés megválaszolása azonban természetesen nem ennyire egyértelmû.

Sokan nem is magát az operációs rendszert, hanem inkább az alkalmazásokat használják. Valójában pedig maguk az alkalmazások azok, amelyek az operációs rendszert használják. A FreeBSD-t úgy alakították ki, hogy az alkalmazások számára egy szilárd és mindentudó környezetet nyújtson. Támogatja a böngészõk, irodai programcsomagok, levelezõ programok, grafikus programok, programozási környezetek, hálózati szerverek széles választékát, és szinte minden mást, amire csak szükségünk lehet. Az ilyen alkalmazások legnagyobb része elérhetõ a Portgyûjteményen keresztül.

Ha viszont olyan alkalmazást kívánunk használni, amely csak bizonyos operációs rendszereken érhetõ el, nem tudjuk magát az operációs rendszert egyszerûen lecserélni alatta. Bizonyos esetekben azonban elõfordulhat, hogy FreeBSD alatt is találunk hozzá hasonló alkalmazásokat. Amikor egy stabil irodai vagy internet szerverre van szükségünk, esetleg egy megbízható munkaállomásra, vagy egyszerûen csak megszakítások nélkül szeretnénk végezni a munkánkat, a FreeBSD-ben igényeinkhez mérten szinte minden megtalálhatunk. A világon rengeteg felhasználó, beleértve a kezdõket és a tapasztalt UNIX® rendszergazdákat egyaránt, asztali operációs rendszerként is a FreeBSD-t használja.

Ha egy másik UNIX® környezetrõl akarunk FreeBSD-re váltani, akkor a legtöbb dolog már ismerõs lehet számunkra. Amennyiben viszont valamilyen grafikus operációs rendszerrõl, például Windows®-ról vagy a Mac OS® valamelyik régebbi változatáról szándékozunk átállni, minden bizonnyal idõt kell majd szánnunk a feladatok UNIX® stílusú megvalósításának megismerésére. Ez a GYIK és a FreeBSD kézikönyv ehhez tökéletes kiindulási alapot biztosít.

1.5. Miért hívják FreeBSD-nek?

- Szabadon (mint "free") felhasználható, akár kereskedelmi célokra is.
- Az operációs rendszer teljes forráskódja bárki által szabadon elérhetõ, minimális megkötésekkel arra vonatkozóan, hogy miként használható és más (kereskedelmi vagy nem kereskedelmi) munkák részeként miként építhetõ be, terjeszthetõ.

- Bárki, akinek fejlesztési vagy hibajavítási javaslata van a rendszerrel kapcsolatban, szabadon benyújthatja azt, amely aztán bekerül a források közé (egy-két nyilvánvaló ellenõrzést követõen).

Érdemes valamint rámutatni, hogy a "szabad" szót az imént két értelemben is használtuk: az egyik jelentése szerint "költségek nélkül", míg a másik jelentése szerint "tetszés szerint". Egy-két *tiltott* dologtól, például azt állítjuk, hogy mi írtuk, eltekintve tényleg bármit csinálhatunk vele.

1.6. Mi a különbség a FreeBSD, a NetBSD, OpenBSD és a többi nyílt forráskódú BSD operációs rendszerek között?

James Howard The BSD Family Tree címmel (angolul) készített egy alapos leírást a különbözõ projektek közti eltérések bemutatására.

1.7. Melyik a FreeBSD legújabb változata?

Jelen pillanatban a FreeBSD fejlesztése két párhuzamos ágon folyik, és mind a kettõbõl készülnek kiadások. A 7.*X* sorozat kiadásai a *7-STABLE* ágból, míg a 8.*X* sorozat kiadásai a *8-STABLE* ágból készülnek.

A 8.0-s kiadás megjelenéséig a 7.*X* sorozat volt a *-STABLE*. A 8.0 kiadás megjelenésével azonban a 7.*X* ág "meghosszabbított támogatást" kapott, és már csak a nagyobb hibákat, például a biztonsági hibákat javítják benne. Az *7-STABLE* ágból még várhatóak további kiadások is, azonban ezt jelenleg már "örökségi" ágnak tekintjük, és a legtöbb munka már a *8-STABLE* részeként jelenik meg.

A 12.0 változat a *8-STABLE* ág legfrissebb kiadása, amely December 11, 2018ban jelent meg. Az *7-STABLE* ágból a 11.2 a legfrissebb kiadás, amely June 28, 2018ban jelent meg.

Ha röviden össze akarjuk foglalni, akkor a *-STABLE* változatokat elsõsorban az internet-szolgáltatók, vállalkozások számára ajánljuk, illetve minden olyan felhasználó számára, aki a legújabb (és minden bizonnyal még instabil) *-CURRENT* pillanatkiadásokhoz viszonyítottan a legkevesebb változtatással kívánnak egy megbízható, stabil verziót használni a rendszerbõl. Ugyan bármelyik ágból készülhetnek, azonban a *-CURRENT* esetében meglehetõsen sok változásra kell felkészülnünk (a *-STABLE* ághoz képest) az egyes kiadások között.

A kiadások néhány havonta készülnek. Mivel a legtöbben ennél pontosabban követik a FreeBSD forrásait (lásd a FreeBSD-CURRENT és FreeBSD-STABLE változatokra vonatkozó kérdéseket), ennél valamire többre van szükségünk, hiszen a források folyamatosan változnak.

A FreeBSD egyes kiadásairól a Kiadások megjelentetését összefoglaló oldalon tájékozódhatunk a FreeBSD honlapján.

1.8. Mi az a FreeBSD-CURRENT?

A FreeBSD-CURRENT az operációs rendszer aktív fejlesztés alatt álló változata, amely idõvel az új FreeBSD-STABLE ággá válik. Ez a változat tulajdonképpen csak a rendszeren dolgozó fejlesztõk és a

megátalkodott hobbifelhasználók számára érdekes. A kézikönyv erre vonatkozó szakaszában olvashatunk részletesebben a *-CURRENT* használatáról.

Ha nem mozgunk otthonosan az operációs rendszerek világában, vagy ha nem tudjuk megmondani a különbséget egy valódi és egy ideiglenes probléma között, akkor nem javasoljuk a FreeBSD-CURRENT használatát. Ez a fejlesztési vonal nagyon gyorsan fejlõdik és néha lefordíthatatlan állapotba kerül. A FreeBSD-CURRENT változat használóitól elvárjuk, hogy képesek legyenek felmérni a felbukkanó problémákat, és közülük csak azokat jelenteni, amelyek valóban hibákat takarnak és nem pedig csak apró "bökkenõk". Ezért a FreeBSD-CURRENT levelezési lista olvasói általában "A make world parancs valami csoportra panaszkodik" típusú kérdéseket általában figyelembe se veszik.

A *-CURRENT* és *-STABLE* ágak aktuális állapotáról minden hónapban pillanatkiadások készülnek. Célunk ezzel:

- A telepítõ legfrissebb változatának tesztelése.
- Idõt és sávszélességet szeretnénk megspórolni a *-CURRENT* vagy *-STABLE* változatok azon felhasználóinak, akik az iméntiek hiányából fakadóan nem tudják naponta frissíteni a rendszerüket.
- Kiindulási pontokat rögzítünk a kód aktuális állapota alapján, ha késõbb netalán valamilyen szörnyûség történne. (Noha a CVS általában védelmet nyújt az ilyen rémisztõ dolgok bekövetkezése ellen.)
- Az összes tesztelendõ újítást és javítást ezen a módon kívánjuk a lehetõ legszélesebb körben elérhetõvé tenni.

Egyik *-CURRENT* pillanatkiadás sem tekinthetõ "hétköznapi felhasználásra alkalmasnak". Ha egy megbízható és széles körben tesztelt rendszerre van szükségünk, akkor vagy maradjunk a kiadásoknál vagy használjuk a *-STABLE* vonalból készült pillanatkiadásokat.

A pillanatkiadások innen érhetõek el.

Minden aktívan fejlesztett ághoz havonta készülnek hivatalos pillanatkiadások. A népszerûbb i386 és amd64 ágakból azonban napi kiadások is elérhetõek a http://snapshots.us.freebsd.org a címen.

1.9. Mit takar a FreeBSD-STABLE?

Amikor a FreeBSD 2.0.5 megjelent, a FreeBSD fejlesztése kettévált. Az egyik ág neve -STABLE, a másiké pedig -CURRENT lett. A *FreeBSD-STABLE* az olyan internet-szolgáltatók és egyéb vállalkozások számára készült, ahol a fejlesztés alatt álló újítások vagy a hirtelen váltások által okozott problémák gyakran nem engedhetõek meg. Ide csak olyan hibajavítások és kisebb módosítások kerülnek, amelyeket alaposan leteszteltek. A *FreeBSD-CURRENT* ezzel szemben a 2.0 megjelenése óta egyetlen, szakadásmentes fejlesztési vonalat képvisel, amely a 12.0-RELEASE és az azon túli kiadások felé halad. Ha többet szeretnénk megtudni a jelenlegi ágak állapotáról és a következõ kiadások ütemezésérõl, akkor ezzel kapcsolatban olvassuk el a FreeBSD Release Engineering címû cikk kiadások leágaztatásáról szóló részét (angolul). Az ágak jelenlegi állapota és a jövõbeni kiadások ütemterve a Kiadások információk oldalán található (angolul).

A 2.2-STABLE ág a 2.2.8 megjelenésével nyugdíjba vonult. A 3-STABLE ág a 3.5.1 mint az utolsó 3.*X*

kiadás megjelenésével ért véget. A 4-STABLE ág a 4.11 mint az utolsó 4.*X* kiadással fejezõdött be. Ezekbe az ágakban a legtöbb esetben már csak biztonsági javításokat végeznek. Az 5-STABLE ág fejlesztése az utolsó 5.*X* kiadás, az 5.5 megjelenésével lezárult. A 6-STABLE ág fejlesztése még folytatódik valameddig, de ez alatt leginkább már csak a biztonsági rések és egyéb komoly problémák javításait kell érteni.

A 12.0-STABLE a jelenleg fejlesztett *-STABLE* ág. A 12.0-STABLE ágból megjelent legfrissebb kiadás a 12.0-RELEASE, amely December 11, 2018ban jelent meg.

A 9-CURRENT a *-CURRENT* ág legfrissebb változata, és ez a FreeBSD következõ generációja. Errõl az ágról a Mi az a FreeBSD-CURRENT? kérdésnél szolgálunk részletesebb információkkal.

1.10. Mikor készülnek FreeBSD kiadások?

A Release Engineering Team <re@FreeBSD.org> átlagosan a FreeBSD egy újabb nagyobb változatát 18 havonta, míg egy kisebb kiadását 8 havonta jelenteti meg. A kiadások dátumát elõre kihirdetik, így a rendszeren dolgozó emberek pontosan tudják, hogy mikorra kell befejezniük a munkájukat és letesztelni azt. Minden kiadást egy tesztelési idõszak elõz meg, ahol megbizonyosodnak róla, hogy az elkészült újítások nem veszélyeztetik az új kiadás megbízhatóságát. A legtöbb felhasználó pontosan ezt a típusú elõvigyázatosságot szereti legjobban a FreeBSD-ben, még annak árán is, hogy a legújabb finomságok bekerülésére még a *-STABLE* ág esetén gyakran sokat kell várni.

A kiadások szerkesztésérõl (valamint a soronkövetkezõ kiadások ütemezésérõl) a FreeBSD honlapján belül ezen az oldalon olvashatunk részletesebben (angolul).

Akik egy kicsivel több izgalomra vágynak, azok részére az elõbb említett, naponta készített bináris pillanatkiadásokat ajánljuk.

1.11. Ki felel a FreeBSD-ért?

A FreeBSD Projektre vonatkozó fontosabb döntéseket, mint például a Projekt haladási irányát vagy hogy vehet részt a forráskód fejlesztésében, egy 9 fõs irányító csoport hozza. Rajtuk kívül még egy több mint 350 fõs fejlesztõi csapat jogosult közvetlenül módosítani a FreeBSD forrásait.

A legtöbb bonyolultabb változtatást általában azonban a megfelelõ levelezési listákon is megvitatják, amiben bárki különösebb korlátozás nélkül részt vehet.

1.12. Honnan lehet a FreeBSD-t beszerezni?

A FreeBSD összes fontosabb kiadása elérhetõ anonim FTP-n keresztül a FreeBSD FTP oldaláról:

- A legfrissebb 8-STABLE kiadás, a 12.0-RELEASE ebbõl a könyvtárból érhetõ el.
- Havonta készülnek pillanatkiadások a -CURRENT és a -STABLE ágakból, de ezek leginkább a legújabb változatot tesztelõk és a fejlesztõk számára fontosak.
- A legfrissebb 7-STABLE kiadás, a 11.2-RELEASE ebbõl a könyvtárból érhetõ el.

Ha a FreeBSD-t CD-n, DVD-n vagy más egyéb telepítõeszközön szeretnénk megkapni, akkor ezzel kapcsolatban nézzük meg a kézikönyvet.

1.13. Hogyan lehet elérni a hibajelentések adatbázisát?

A felhasználók kéréseit tartalmazó hibajelentések adatbázisát a honlap webes hibajelentésekkel foglalkozó felületén keresztül érhetjük el.

A send-pr(1) parancs segítségével tudunk e-mailen keresztül hibajelentéseket és egyéb változtatási kéréseket küldeni. Emellett még böngészõ segítségével is tudunk hibajelentéseket küldeni a honlap webes hibabejelentõ felületén.

Mielõtt beküldenénk egy hibajelentést, olvassuk el a Writing FreeBSD Problem Reports címû cikket (angolul), amelybõl megtudhatjuk, hogyan készítsünk jól hasznosítható hibajelentéseket.

1.14. Honnan tudhatunk meg még többet?

Nézzük meg a FreeBSD Projekt honlapjáról elérhetõ dokumentációkat.

Chapter 2. Dokumentációs és támogatás

2.1. Milyen jó könyvek szólnak a FreeBSD-rõl?

A Projekt igen széles körû dokumentációval rendelkezik, amely a következõ linkrõl érhetõ el: http://www.FreeBSD.org/docs/. Emellett a GYIK végén szereplõ, valamint a kézikönyvben található irodalomjegyzék tartalmazza az ajánlott könyveket.

2.2. A dokumentáció elérhetõ más formátumokban is, például szöveges (ASCII) állományban vagy PostScript®-ben?

Igen. A dokumentáció több különbözõ állomány- és tömörítési formátumban elérhetõ az FreeBSD FTP oldalán belül a /pub/FreeBSD/doc/ könyvtárból.

A dokumentációt több különbözõ módon osztályozhatjuk. Többek közt:

- A dokumentum neve alapján, például `faq` (GYIK), vagy `handbook` (kézikönyv).
- A dokumentum nyelv és karakterkódolása alapján. Ezeket a FreeBSD rendszerekben, a /usr/shared/locale könyvtárban megtalálható nyelvi beállítások nevei szerint adjuk meg. Jelenleg a következõ nyelveken és kódolásokban érhetõ el a dokumentáció:

Név	Leírás
`en_US.ISO8859-1`	Angol (Egyesült Államok)
`bn_BD.ISO10646-1`	Bengáli vagy bangla (Banglades)
`da_DK.ISO8859-1`	Dán (Dánia)
`de_DE.ISO8859-1`	Német (Németország)
`el_GR.ISO8859-7`	Görög (Görögország)
`es_ES.ISO8859-1`	Spanyol (Spanyolország)
`fr_FR.ISO8859-1`	Francia (Franciaország)
`hu_HU.ISO8859-2`	Magyar (Magyarország)
`it_IT.ISO8859-15`	Olasz (Olaszország)
`ja_JP.eucJP`	Japán (Japán, EUC kódolás)
`mn_MN.UTF-8`	Mongol (Mongólia, UTF-8 kódolás)
`nl_NL.ISO8859-1`	Holland (Hollandia)
`no_NO.ISO8859-1`	Norvég (Norvégia)
`pl_PL.ISO8859-2`	Lengyel (Lengyelország)
`pt_BR.ISO8859-1`	Portugál (Brazília)
`ru_RU.KOI8-R`	Orosz (Oroszország, KOI8-R kódolás)

Név	Leírás
sr_YU.ISO8859-2	Szerb (Szerbia)
tr_TR.ISO8859-9	Török (Törökország)
zh_CN.GB2312	Egyszerûsített kínai (Kína, GB2312 kódolás)
zh_TW.Big5	Hagyományos kínai (Tajvan, Big5 kódolás)

Nem mindegyik dokumentum érthetõ el mindegyik nyelven.

- A dokumentum formátuma alapján. A dokumentumok több különbözõ formátumban állnak rendelkezésre. Mindegyik formátum használatának megvannak az elõnyei és hátrányai. Egyes formátumok inkább az interneten keresztüli olvasgatásra megfelelõek, mások pedig nyomtatott formában nyújtanak esztétikus hatást. A több különbözõ formátumnak köszönhetõen az olvasók igényeik szerint el tudják olvasni a dokumentáció különbözõ részeit akár a képernyõn, akár papíron. Jelenleg a következõ formátumokban érhetõek el a dokumentumok:

Formátum	Leírás
html-split	Kis méretû, hiperhivatkozásokkal ellátott HTML állományok gyûjteménye
html	Egyetlen óriási, az egész dokumentumot tartalmazó HTML állomány
pdf	Az Adobe-féle Portable Document Format
ps	PostScript®
rtf	A Microsoft Rich Text formátuma
txt	Egyszerû szöveges állomány

Amikor egy ilyen dokumentumot betöltünk a Wordbe, akkor az oldalszámok maguktól nem frissülnek. Ehhez a dokumentum betöltése után nyomjuk le a Ctrl + A, Ctrl + End, F9 billentyûket.

- A tömörítés és csomagolás típusa alapján. Ezek közül jelenleg hármat használunk.

  a. Ahol a formátum html-split, ott az állományokat a tar(1) segítségével csomagoltuk össze. Az így keletkezõ .tar állományt ezek után az alábbi részben szereplõ tömörítési megoldásokkal tömörítettük.

  b. Az összes többi formátum esetén csak egyetlen állomány keletkezik, amelynek a neve típus.formátum (tehát például article.pdf, book.html és így tovább).

  Ezeket az állományokat azután két tömörítési eljárással tömörítjük.

Eljárás	Leírás
zip	A zip formátum. FreeBSD alatt ezt úgy tudjuk kitömöríteni, ha elõször telepítjük a archivers/unzip portot.

Eljárás	Leírás
bz2	A bzip2 formátum. Nem olyan elterjedt, mint a zip, de általában kisebb méretû állományokat készít. Ilyen állományokat akkor tudunk kitömöríteni, ha telepítjük a archivers/bzip2 portot.

Ennek megfelelõen tehát a kézikönyv bzip2-vel tömörített PostScript® változata a handbook/ könyvtáron belül book.ps.bz2 néven található.

Miután kiválasztottuk a számunkra megfelelõ letöltendõ formátumot és tömörítési módszert, magunknak kell letölteni a kiválasztott tömörített állományokat, majd kibontani ezeket és átmásolni a megfelelõ helyre.

Például, ha a GYIK fejezetekre darabolt, bzip2(1) segítségével tömörített változata a doc/en_US.ISO8859-1/books/faq/book.html-split.tar.bz2 állományban található meg. A letöltéséhez és kibontásához a következõket kell tennünk:

```
# fetch ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/en_US.ISO8859-1/books/faq/book.html-
split.tar.bz2
# bzip2 -d book.html-split.tar.bz2
# tar xvf book.html-split.tar
```

A mûvelet befejezõdésével kapunk néhány .html kiterjesztésû állományt. Ezek közül az egyik neve index.html, ebben található a tartalomjegyzék, a bevezetés és a dokumentum többi részére mutató hivatkozások. Ezeket az állományokat kell szükség szerint átmásolnunk vagy átmozgatnunk a megfelelõ helyre.

2.3. Hol található információ a FreeBSD levelezési listáiról?

Az összes velük kapcsolatos információt a kézikönyv levelezési listákról szóló részében találjuk.

2.4. Milyen FreeBSD hírcsoportok léteznek?

Az összes rájuk vonatkozó információt a kézikönyv hírcsoportokról szóló részében találjuk meg.

2.5. Vannak FreeBSD-s IRC (Internet Relay Chat) csatornák?

Igen, a legtöbb nagyobb IRC hálózaton található FreeBSD-vel foglalkozó csatorna:

- Az EFNet hálózaton található #FreeBSD csatorna lényegében egy FreeBSD-vel foglalkozó fórum, de itt ne nagyon próbálkozzunk segítséget kérni a többiektõl, ha netalán lusták lennénk elolvasni a man oldalakat vagy éppen kutatunk valamit. Ez a hely elsõsorban csevegésre szolgál,

ahol mindenféle téma felmerül, a szextõl kezdve a sportokon keresztül a nukleáris fegyverekig éppen úgy, ahogy a FreeBSD-rõl is. Mi szóltunk elõre! A szerver a irc.efnet.org címen érhetõ el.

- Az EFNet hálózaton található #FreeBSDhelp csatorna kifejezetten a FreeBSD felhasználók megsegítését veszi célba. Az itt levõk sokkal szívesebben válaszolnak a kérdéseinkre, mint a #FreeBSD csatornán.
- A Freenode hálózaton található FreeBSD csatornán mindig sokan vannak, itt bármilyen témában kérhetünk segítséget. A beszélgetések idõnként ugyan kifutnak a szigorú szakmai témákból, de a FreeBSD-vel kapcsolatos kérdések itt mindig elsõbbséget élveznek. Szívesen segítünk bárkinek, és lehetõség szerint igyekszünk a kézikönyv megfelelõ részeire hivatkozni, vagy adni valamilyen útmutatást arra vonatkozóan, hogy merre tájékozódhatunk részletesebben a problémánkkal kapcsolatban. Ez alapvetõen egy angol nyelvû csatorna, habár a világ minden tájáról érkeznek tagjaink. Ha az anyanyelvünkön szeretnénk inkább csevegni, akkor elõször tegyük fel a kérdésünket angolul, aztán próbálkozzunk a megfelelõfreebsd-nyelv csatornán.
- A DALNET hálózaton található #FreeBSD csatorna az Egyesült Államokból a irc.dal.net szerveren, Európából pedig az irc.eu.dal.net szerveren keresztül érhetõ el.
- A DALNET hálózaton található #FreeBSDHelp csatorna az Egyesült Államokból a irc.dal.net szerveren, Európából pedig a irc.eu.dal.net szerveren keresztül érhetõ el.
- Az UNDERNET hálózaton található #FreeBSD csatorna az Egyesült Államokból a us.undernet.org, Európából pedig a eu.undernet.org szerveren keresztül érhetõ el. Mivel ez a csatornát leginkább segítségnyújtásra tartjuk fenn, készüljünk fel arra, hogy a hivatkozott dokumentumokat is el kell olvasnunk.
- A RUSNET hálózaton található #FreeBSD csatorna az oroszul beszélõ FreeBSD felhasználók számára igyekszik segítséget nyújtani. Emellett viszont remek hely a nem szakmai jellegû témák megvitatásához is.
- A Freenode hálózaton található #bsdchat csatorna a hagyományos kínai (UTF-8 kódolású) nyelvet beszélõ FreeBSD felhasználókat igyekszik segíteni. A nem szakmai jellegû témák részére is egy remek hely.

Az említett csatornák mindegyike egymástól független, és nem állnak egymással kapcsolatban. Sõt, még a csevegési stílusuk is eltérõ, ezért érdemes a saját stílusunkhoz leginkább illeszkedõt megkeresni. Mint ahogy az *összes* IRC csatorna esetében elõfordul, itt is könnyedén érhetnek bennünket személyes sértések vagy egyszerûen csak sok szóbeli sárdobálást láthatunk (mivel jóval több az ilyen helyeken a balga ifjú, mint a higgadtabb idõs) - ezekkel ne is törõdjünk!

2.6. Hol kaphatok kereskedelmi szintû FreeBSD tréninget és támogatást?

A FreeBSD Mall is nyújt keresdelmi támogatást a FreeBSD-hez. Errõl a honlapjunkon tudhatunk meg többet.

A BSD Certification Group, Inc. DragonFly BSD, FreeBSD, NetBSD és OpenBSD rendszerekhez ad rendszergazdai képesítéseket. Amennyiben érdekel minket, látogassunk el a honlapjukra.

Kérünk minden olyan további szervezetet, amely tréninget vagy támogatást kíván nyújtani a Projektnek, hogy jelentkezzenek és felvesszük õket a listánkra!

Chapter 3. Telepítés

3.1. Milyen állományokat kell letöltenünk a FreeBSD telepítéséhez?

Ehhez a következõ három floppy image-re lesz alapvetõen szükségünk: floppies/boot.flp, floppies/kern1.flp és floppies/kern2.flp. Ezeket az image-eket az `fdimage` vagy dd(1) segédprogramokkal kell rámásolnunk lemezekre.

Ha magukat a terjesztéseket akarjuk letölteni (mert például egy DOS típusú állományrendszerrõl akarunk telepíteni), akkor az alábbi terjesztéseket kell beszereznünk:

- base/
- manpages/
- compat*/
- doc/
- src/ssys.*

A teljes folyamatot, valamint a telepítéssel kapcsolatos általános tudnivalókat valamivel bõvebben a kézikönyv FreeBSD telepítésével foglalkozó részébõl ismerhetjük meg.

3.2. Mit tegyünk, ha a floppy image-ek nem férnek rá egyetlen lemezre?

Egy 3,5 colos (1,44 MB kapacitású) lemezen 1 474 560 byte-nyi adat fér el. A rendszerindításhoz használt image mérete is pontosan 1 474 560 byte.

A rendszerindító lemezek elõkészítése során elkövetett hibák általában a következõk:

- Amikor az image-eket FTP-n keresztül töltjük le, elfelejtünk *bináris* (binary) átviteli módot használni.

  Egyes FTP kliensek alapértelmezés szerint *szöveges* (ascii) módban viszik át az állományokat, és ennek során megpróbálják a sorvége karaktereket az adott operációs rendszer konvenciói szerint átalakítani. Ilyenkor szinte kétségtelen, hogy ezzel tönkreteszik az image-et. Ezért ne felejtsük el ellenõrizni a letöltött image-eket: ha a méretük nem egyezik meg *pontosan* a szerveren levõ változatukéval, akkor gyaníthatóan a letöltés közben történt velük valami.

  Megoldás: miután csatlakoztunk a szerverhez, de még mielõtt elkezdük volna a letöltést, az FTP kliens parancssorában gépeljük be, hogy *binary*.

- Az image lemezre másolása a DOS `copy` parancsának (vagy hasonló grafikus eszközök) használatával.

  A `copy` és a hozzá hasonló programok nem használhatóak erre a célra, mivel az image-eket közvetlenül a rendszeindításhoz hozták létre. Ennek megfelelõen az egyes image-ek a lemezek

teljes tartalmát sávról sávra tartalmazzák, és így nem hétköznapi állományként kell velük bánni. Ezeket a floppykra alacsonyszintû eszközök (például az `fdimage` vagy `rawrite`) segítségével, "nyers" módban kell felvinni, ahogy azt a FreeBSD telepítését leíró útmutatóban is olvashatjuk.

3.3. Hol található leírás a FreeBSD telepítéséröl?

A telepítés részletes leírása a kézikönyv FreeBSD telepítéséröl szóló részében olvasható.

3.4. Mire van szükség a FreeBSD használatához?

A FreeBSD használatához egy 486-os vagy jobb processzorral rendelkezõ számítógépre, 24 MB vagy annál több memóriára, és legalább 150 MB tárhelyre lesz szükségünk.

A FreeBSD összes változata képes futni szinte bármilyen olcsó MDA típusú grafikus kártyával, de az Xorg használatához már VGA vagy annál jobb videokártya szükségeltetik.

Lásd Hardverkompatibilitás.

3.5. Hogyan lehet saját telepítõfloppyt készíteni?

Jelen pillanatban ennek nincs *egyszerû* módja. Minden egyes kiadáshoz tartoznak telepítõfloppyk, használjuk ezeket.

Ha egy módosított kiadást akarunk készíteni, kövessük a(z angol nyelvû) Release Engineering cikk útmutatásait.

3.6. Windows® mellé is telepíthetõ FreeBSD?

Elõször telepítsük a Windows®t, majd a FreeBSD-t. A FreeBSD boot managere ekkor képes lesz a Windows® és a FreeBSD indítására is. Vigyázzunk, mert ha a Windows®t telepítjük fel másodikként, akkor az minden figyelmeztetés nélkül durván felülírja az aktuális boot managert. Ha ezt tapasztaljuk, akkor olvassuk el a következõ szakaszt.

3.7. A Windows® letörölte a boot managert! Hogyan lehet visszaállítani?

A FreeBSD-hez tartozó boot managert háromféleképpen tudjuk újratelepíteni:

- Indítsuk el a DOS-t, lépjünk be a FreeBSD terjesztéshez tartozó tools könyvtárba és keressük meg a bootinst.exe nevû állományt. Indítsuk el a következõ módon:

  ```
  ...\TOOLS> bootinst.exe boot.bin
  ```

  Ekkor a boot manager visszakerül a helyére.

- Használjuk a FreeBSD-hez létrehozott rendszerindító lemezeket, és a telepítõben válasszuk a Custom (Egyéni telepítés) menüpontot, majd azon belül válasszuk a Partition (Partíció) pontot. Itt válasszuk ki azt a meghajtót, ahol korábban a boot managerünk volt (ez valószínûleg a felsorolásban az elsõ lesz) és amikor belépünk a partíciószerkesztõbe, akkor egybõl válasszuk a `Write` (`W`) opciót (tehát ne változtassunk semmit). Ez megerõsítést fog kérni, amire válasszuk a **[yes]** gombot, és amikor a boot manager kiválasztása rész jelenik meg, válasszuk a FreeBSD Boot Manager pontot. Ezzel a boot manager újra a lemezre íródik. Miután ezzel végeztünk, lépjünk ki a telepítõbõl és indítsuk újra a rendszerünket a megszokott módon.
- Indítsuk a rendszerünket a FreeBSD rendszerindító lemezérõl (vagy CD-jérõl), majd válasszuk a telepítõben a Fixit (Javítás) menüpontot. Ezután válasszuk a javítófloppy vagy a(z "élõ" állományrendszerrel rendelkezõ) 2. CD használatát, majd lépjünk be a javításhoz elindított parancsértelmezõbe. Ezt követõen adjuk ki az alábbi parancsot:

  ```
  Fixit# fdisk -B -b /boot/boot0 eszköz
  ```

  A parancsban az *eszköz* helyére annak az eszköznek a nevét adjuk meg, amelyrõl a rendszert szoktuk indítani, például ad0 (az elsõ IDE-lemez), ad4 (az elsõ IDE-lemez valamelyik vezérlõn), da0 (az elsõ SCSI-lemez) stb.

3.8. Az A, T és X sorozatú IBM Thinkpad laptopok lefagynak a FreeBSD telepítése utáni elsõ indulásuk során. Hogy lehet ezen segíteni?

Ezeken a gépeken az IBM BIOS-ának egy korai hibás változata található, amely a FreeBSD által használt partíciókat tévesen "suspend-to-disk" típusú partícióknak tekinti. Ennek következtében amikor a BIOS megpróbálja értelmezni a FreeBSD által létrehozott partíciót, megakad.

Az IBM szerint az alábbi típus/BIOS változatokban található meg ez a hiba.

Típus	BIOS
T20	IYET49WW vagy késõbbi
T21	KZET22WW vagy késõbbi
A20p	IVET62WW vagy késõbbi
A20m	IWET54WW vagy késõbbi
A21p	KYET27WW vagy késõbbi
A21m	KXET24WW vagy késõbbi
A21e	KUET30WW

Úgy értesültünk, hogy az IBM BIOS-ok késõbbi változataiban ismét felbukkant ez a típusú hiba. Ebben az üzenetben Jacques Vidrine <nectar@FreeBSD.org> a FreeBSD laptop computer levelezési lista tagjainak egy olyan módszert mutat be, ami segíthet, ha az újabb típusú IBM laptopunk nem tudja elindítani a FreeBSD-t, és így váltani tudunk a BIOS elõzõ vagy következõ verziójára.

Ha régebbi típusú BIOS-szal rendelkezünk és a frissítés nem megoldható, akkor a FreeBSD-t telepíthetjük úgy is, hogy megváltoztatjuk a FreeBSD által használt partíció azonosítóját és egy olyan rendszerindító blokkot telepítünk, amelyik képes ezt kezelni.

Ehhez elõször is a gépet egy olyan állapotba kell visszahoznunk, ahol már túl tudunk jutni a rendszerindító képernyõn. Ezt úgy tudjuk elérni, ha nem engedjük, hogy a gép indulása közben észrevegye az elsõdleges lemezen található FreeBSD partíciót. Erre az egyik lehetséges megoldás, ha a gépbõl ideiglenesen eltávolítjuk a merevlemezt és átrakjuk egy régebbi ThinkPadba (például egy ThinkPad 600-as típusba) vagy a megfelelõ átalakító használatával az asztali számítógépünkbe. Miután ezzel megvagyunk, töröljük le a FreeBSD partícióját és tegyük vissza a lemezt. Ekkor a ThinkPad újból mûködõképes lesz.

Ezt követõen az alábbi utasításokat követve tudjuk telepíteni a FreeBSD-t:

1. Töltsük le a boot1 és boot2 állományokat a http://people.FreeBSD.org/~bmah/ThinkPad/ címrõl. Olyan helyre tegyük ezeket, ahol késõbb is még el tudjuk érni.
2. A megszokott módon telepítsük a FreeBSD-t a ThinkPadre. Ilyenkor *ne* használjuk a `Veszélyesen dedikált` (Dangerously Dedicated) módot. A telepítés befejezése után *ne* indítsuk újra a gépet.
3. Váltsunk át a vészhelyzetekben használatos parancsértelmezõre ("Emergency Holographic Shell", `Alt` + `F4`) vagy indítsuk el egy javításhoz használt ("fixit") parancsértelmezõt.
4. Az fdisk(8) segítségével változtassuk meg a FreeBSD-s partíció azonosítóját a `165` értékrõl a `166` értékre (ezt a típust az OpenBSD használja).
5. Másoljuk át az imént letöltött boot1 és boot2 állományokat a helyi állományrendszerre.
6. A disklabel(8) segítségével rögzítsük a boot1 és boot2 tartalmát a FreeBSD slice-unkra.

   ```
   # disklabel -B -b boot1 -s boot2 ad0sn
   ```

   ahol az *n* annak a slice-nak a sorszáma, ahová a FreeBSD-t telepítettük.
7. Indítsuk újra a gépet. A rendszerindító parancssorban ekkora megjelenik az `OpenBSD` indításának lehetõsége. Ezen keresztül tudjuk a FreeBSD-t elindítani.

A kedves Olvasónak meghagytuk azt az esetet, amikor ugyanezen a konfiguráción OpenBSD és FreeBSD rendszereket akarunk egyszerre használni.

3.9. Lehet telepíteni hibás szektorokat tartalmazó lemezre is?

Igen, ez lehetséges, de egyáltalán nem ajánlott.

Manapság ha egy IDE-meghajtón hibás szektorokat találunk, akkor az arra utal, hogy hamarosan tönkremegy (a meghajtó belsõ átképezõ funkciói már képesek megbirkózni a rossz szektorok növekvõ számával, ami arra enged következtetni, hogy a lemez felülete jelentõs mértékben sérült).

Ezért inkább egy új merevlemezes meghajtó vásárlását javasoljuk.

Ha hibás SCSI-meghajtónk van, ezt a választ olvassuk el.

3.10. Furcsa dolgok történnek a telepítõfloppy használata közben! Mi okozhatja?

Ha olyan furcsa dolgokkal találkozunk a telepítõfloppy használata során, mint például a lemez állandó darálása vagy a rendszer váratlan újraindulása, akkor a következõ három kérdést érdemes feltennünk magunknak:

1. Biztos, hogy új, frissen formázott, teljesen hibamentes floppykat használunk (tehát olyanokat, amelyeket egy frissen bontott dobozból vettünk ki, és nem olyanokat, amelyeket valamelyik magazin mellékletébõl szedtük ki vagy éppen három évig az ágy alatt tároltunk)?
2. Biztos, hogy bináris (vagy image) módban töltöttük le a lemezek image-eit? (Ne szégyelljük, mindenki életében legalább egyszer töltött már le véletlenül bináris állományt szöveges formátumban!)
3. Windows® 95 vagy Windows® 98 alatt DOS módban használtuk az `fdimage` vagy `rawrite` parancsot? Ezek az operációs rendszerek általában nem férnek össze az olyan programokkal, amelyek közvetlenül a hardverrel akarnak kommunikálni, amire a lemezek írásához is szükség van. Ez a probléma leginkább akkor merülhet fel, amikor a grafikus felületen belül egy DOS ablakban futtatjuk ezeket a programokat.

Kaptunk olyan visszajelzést is, hogy gondjaink lehetnek, ha getenv(3)-pel töltjük le a rendszerindító lemezeket, ezért lehetõség szerint igyekezzünk más FTP klienst használni.

3.11. ATAPI CD-meghajtóról indult a rendszer, de a telepítõ szerint nem található semmilyen CD-meghajtó. Hova tûnt?

Ezt a problémát általában egy rosszul beállított CD-meghajtó okozza. A CD-meghajtó rengeteg számítógépben a másodlagos IDE-vezérlõ slave (szolga) portján található, a master (mester) port használata nélkül. Ez az ATAPI specifikációi szerint nem szabályos, de a Windows® ezzel különösebben nem törõdik, a BIOS pedig egyszerûen figyelmen kívül hagyja a rendszer indítása során. Ezért képes a BIOS ilyenkor látni a CD-meghajtót, és ezért nem képes a FreeBSD teljes telepítésnél használni.

Ezen úgy tudunk segíteni, ha a CD-meghajtónkat az IDE-vezérlõn átállítjuk masterre, vagy arra az IDE-vezérlõre teszünk egy master eszközt.

3.12. PLIP (Parallel Line IP) használatával lehet laptopra telepíteni?

Igen. Ehhez csupán egy szabványos Laplink-kábel kell. Amennyiben szükséges, a párhuzamos vonali hálózatkezelés beállításához olvassuk el kézikönyv PLIP-rõl szóló részét.

3.13. A lemezmeghajtók esetében milyen geometriai beállításokat érdemes használni?

A lemez "geometriája" alatt a lemezen található cilinderek, fejek és a sávonkénti szektorok számát értjük. Ezt a továbbiakban csak CHS-értéknek nevezzük (mint Cylinder/Head/Sector). Ebbõl állapítja meg a PC-s BIOS, hogy a lemezen honnan kell olvasnia és hova kell írnia.

Ez rengeteg félreértést okoz az újdonsült rendszergazdák számára. Elõször is megemlítenénk, hogy egy SCSI-lemez *fizikai* geometriája ebben az esetben teljesen lényegtelen, mivel a FreeBSD lemezblokkokban gondolkozik. Igazából nem létezik "a" fizikai geometria fogalma, ugyanis a szektorok sûrûsége a lemezen felületén belül sem állandó. Amit a gyártók általában "fizikai geometriának" hívnak, az általában az a geometria, amely a legkevesebb helyveszteséggel jár. Az IDE-lemezek esetében a FreeBSD ugyan CHS-értékekkel dolgozik, de ezt minden modernebb meghajtó legbelül blokkhivatkozásokká alakítja.

Egyedül tehát a *logikai* geometria számít. Ez a válasz, amikor a BIOS megkérdezi a meghajtónkat: "Mik a geometriai beállításaid?", és ennek felhasználásával kommunikál vele a késõbbiekben. Mivel a FreeBSD is ezt az értéket használja fel a rendszer indításánál, fontos, hogy jól adjuk meg. Ez különösen abban az esetben számít, amikor több operációs rendszer is található a lemezen, hiszen mindegyiküknek azonos geometriai beállításokat kell használniuk. Ellenkezõ esetben komoly gondok léphetnek fel a rendszer indítása során!

A SCSI-lemezek esetében a beállítandó geometria értéke attól függ, hogy a vezérlõn használjuk-e a bõvített fordítás támogatását (extended translation support, amelyet gyakran csak úgy neveznek, hogy "Support for DOS disks >1GB" vagy ehhez hasonlóan). Ha ezt letiltottuk, akkor használjuk az *N* cilinder, 64 fej és 32 szektor sávonkénti felírást, ahol *N* a lemez MB-okban számított mérete. Így például egy 2 GB méretû lemez geometriai beállítása 2048 cilinder, 64 fej és 32 szektor sávonként.

Ha viszont *engedélyeztük* (ami gyakran elõfordul, mivel így lehet az MS-DOS® bizonyos korlátozásait megkerülni) és a lemez kapacitása 1 GB-nál több, adjunk meg *M* cilindert, 255 fejet, 63 (és *nem* 64) szektort sávonként, ahol az *M* a lemez MB-okban mért kapacitása osztva 7,844238-al (!). Tehát az iménti példában is említett 2 GB-os meghajtó esetében 261 cilindert, 255 fejet és sávonként 63 szektort kapunk.

Ha nem lennénk benne biztosak, vagy a FreeBSD-nek a telepítés közben nem sikerül megállapítania a lemez geometriai beállításait, mi magunk is könnyen meg tudjuk határozni, ha készítünk egy kis méretû DOS partíciót a lemezen. A BIOS ekkor észlelni fogja a megfelelõ geometriai beállításokat, és ha már nincs rá tovább szükségünk, akkor a partíciószerkesztõben nyugodtan törölhetjük. Hálózati kártyák és hasonló hardverek programozásához azonban még a késõbbiekben hasznos lehet.

Használhatjuk viszont a FreeBSD-hez mellékelt pfdisk.exe segédprogramot is. Ezt a FreeBSD CD vagy a FreeBSD FTP oldalainak tools könyvtárában találhatjuk meg. Ennek a programnak a segítségével ki tudjuk deríteni, hogy a lemezen levõ többi operációs rendszer milyen geometriai beállításokat használ. Az így kapott értékeket fel tudjuk használni a partíciószerkesztõben.

3.14. Van valamilyen korlátozás a lemezek felosztására vonatkozóan?

Igen. A rendszerindításhoz használt (gyökér)partíciónak az 1024. cilinder alatt kell kezdõdnie, mivel a BIOS csak így képes betölteni onnan a rendszermagot. (Ez a korlátozás a PC-s BIOS-ok miatt van, nem a FreeBSD miatt.)

A SCSI-lemezek esetében ez általában azt jelenti, hogy rendszerindításhoz használt partíciónak az elsõ 1024 MB alatt kell kezdõdnie (vagy az elsõ 4096 MB alatt, ha a bõvített fordítást is engedélyeztük - lásd az elõzõ kérdést). Az IDE-lemezek esetében ez 504 MB-nak felel meg.

3.15. A FreeBSD kompatibilis valamilyen disk managerrel?

A FreeBSD felismeri az Ontrack Disk Managert és figyelembe veszi. A többi disk managert nem támogatja.

Ha egyedül csak a FreeBSD-t akarjuk használni, akkor nincs szükségünk disk managerre. Egyszerûen csak állítsunk be egy akkora méretû lemezt, amivel a BIOS képes még megbirkózni (a határ általában 504 MB) és majd a FreeBSD kideríti, hogy valójában mennyi hely áll a rendelkezésére. Ha régebbi gyártmányú merevlemezünk van MFM-vezérlõvel, akkor a FreeBSD-nek konkrétan meg kell mondanunk, hogy mennyi cilindert használhat.

Ha a FreeBSD mellett más operációs rendszereket akarunk használni, akkor ezt disk manager nélkül is megtehetjük. Egyedül arra kell vigyáznunk, hogy a FreeBSD indításához használt partíció és a másik operációs rendszer slice-a az elsõ 1024 cilinder alatt kezdõdjön. Ha nagyon körültekintõek akarunk lenni, akkor erre a célra egy 20 MB méretû rendszerindító partíció tökéletesen megfelel.

3.16. Amikor a FreeBSD-t telepítése után elõször elindul, akkor egy Hiányzó operációs rendszer vagy egy Missing Operating System hiba jelenik meg. Mi történt?

Ez általában akkor fordul elõ, amikor a FreeBSD és a DOS vagy más operációs rendszerek nem értenek egyet a lemez geometriai beállításaiban. Telepítsük újra a FreeBSD-t és ezúttal figyelmesen kövessük a fentebb adott utasításokat!

3.17. Miért nem lehet továbblépni a boot manager F? menüjénél?

Ez az elõbbi kérdéssel kapcsolatos probléma egy másik tünete: a BIOS és a FreeBSD által használt geometriai beállítások nem egyeznek! Amennyiben a vezérlõ vagy a BIOS támogatja a cilinderek fordítását (amelyet gyakran ">1GB driver support" néven találhatunk meg), akkor próbáljuk meg

átállítani és így újratelepíteni a FreeBSD-t.

3.18. Az összes forrást telepíteni kell?

Alapvetõen nem. Ettõl függetlenül azonban javasoljuk legalább a `base` források telepítését, ahol számos olyan állomány megtalálható, amelyekre a késõbbiekben még hivatkozni fogunk, valamint a `sys` (rendszermag) források telepítését, amelyben a rendszermag forrásai találhatóak. A rendszeren belül azonban a mûködéshez semmi sem igényli közvetlenül a források jelenlétét, egyedül talán a rendszermag beállítását végzõ config(8) program. A rendszermag forrásainak kivételével a rendszerben a fordítás menetét úgy építettük fel, hogy akár egy írásvédett módon csatlakoztatott NFS állományrendszerrõl is képes legyen dolgozni (a rendszermag forrásaira vonatkozó megszorítások miatt azonban azt javasoljuk, hogy ezt közvetlenül ne a /usr/src könyvtárba csatlakoztassuk, hanem egy másik helyre, ahol aztán szimbolikus linkek segítségével másoljuk le a forráskód könyvtárszerkezetének legfelsõ szintjét).

Ha kéznél vannak a források és tisztában vagyunk a rendszerfordítás folyamatával, akkor a késõbbiekben sokkal könnyebben tudjuk a FreeBSD rendszerünket frissíteni.

A források egyes részeinek kiválasztásához lépjünk be a telepítõprogram Custom (Egyéni telepítés), majd a Distributions (Terjesztések) menübe.

3.19. Kell rendszermagot fordítani?

Egy új rendszermag fordítása korábban fontos része volt a FreeBSD telepítésének, de a legújabb kiadások már kihasználják a rendszermag beállításának sokkal baratságosabb módszereit is. A FreeBSD 5.X és az azt követõ változatokban már a betöltõbõl könnyen be tudjuk állítani a rendszermagot a beépített "hints" (eszközökre vonatkozó útmutatások) módszere által felkínált rugalmasabb lehetõségeknek köszönhetõen.

Egy új rendszermag készítése viszont olyan esetekben még továbbra is hasznos lehet, amikor csak azokat a meghajtókat akarjuk megtartani benne, amelyekre ténylegesen szükségünk van. Ezzel többnyire memóriát tudunk megspórolni, habár a legtöbb rendszer esetében erre igazából nincs szükségünk.

3.20. A jelszavak tárolására használható-e DES, Blowfish vagy MD5, és ha igen, akkor hogyan lehet megadni?

A FreeBSD alapértelmezés szerint *MD5*-alapú jelszavakat használ. Ezeket a *DES* algoritmuson alapuló hagyományos UNIX®-os jelszavaknál sokkal megbízhatóbbnak tartják. A DES formátum természetesen továbbra is elérhetõ olyan esetekben, amikor a kevésbé biztonságos jelszavakat használó régi operációs rendszerekkel akarunk együttmûködni. Emellett a FreeBSD-ben lehetõségünk van a sokkal biztonságosabb Blowfish jelszóformátum használatára is. Az új jelszavak formátumát az /etc/login.conf állományban található `passwd_format` bejelentkezési tulajdonság adja meg, amelynek értéke `des`, `blf` (amennyiben elérhetõ), illetve `md5` lehet. A bejelentkezési tulajdonságokkal kapcsolatban a login.conf(5) man oldalt érdemes elolvasni.

3.21. A rendszerindító lemez elõször elindul, de aztán miért akad meg a Probing Devices... képernyõn?

Ha a rendszerünkhöz IDE-s Zip® vagy Jaz® meghajtót csatlakoztattunk, akkor próbálkozzunk újra az eltávolítása után. A rendszerindító floppy ugyanis hajlamos összekeverni a meghajtókat. A rendszer telepítése után természetesen újra csatlakoztathatjuk a meghajtót. Ezt remélhetõleg egy következõ verzióban már kijavítják.

3.22. A rendszer telepítését követõ újraindítás után miért jelenik meg a panic: can't mount root hibaüzenet?

Ez a hiba a rendszerindító blokk és a rendszermag közti félreértésbõl, a lemezes eszközök helytelen kezelésébõl fakad. Ilyen hibát általában olyan rendszerekben kapunk, ahol két masternek beállított IDE-lemez található vagy ha az egyes IDE-vezérlõkre csak egy-egy eszközt csatlakoztattunk és a FreeBSD-t a másodlagos IDE-vezérlõre kapcsolódó lemezre telepítettük. Ekkor a rendszerindító blokk szerint a rendszert az ad0 (de a BIOS-ban a második) lemezre telepítettük, miközben a rendszermag szerint ez a másodlagos IDE-vezérlõn elhelyezkedõ elsõ lemez, az ad2. Az eszközök felkutatása után a rendszermag megpróbálja a rendszerindító blokk által nyilvántartott eszközrõl, az ad0 lemezrõl csatlakoztatni a rendszerindító partíciót, ami viszont számára a ad2 eszköz lesz, így ez a próbálkozása meghiúsul.

Ezt a félreértést a következõ módokon lehet helyretenni:

1. Indítsuk újra a rendszert és nyomjuk le az `Enter` billentyût, amikor a `Booting kernel in 10 seconds; hit [Enter] to interrupt` szöveg megjelenik. Ezzel a rendszerbetöltõ parancssorába kerülünk.

   Ezután gépeljük be a `set root_disk_unit="lemezszám"` sort. Itt a *lemezszám* értéke `0` lesz, ha a FreeBSD-t az elsõdleges IDE-vezérlõ master portján levõ merevlemezre telepítettük, `1`, ha az elsõdleges IDE-vezérlõ slave portjára, `2`, ha a másodlagos IDE-vezérlõ master portjára, és végül `3`, ha a másodlagos IDE-vezérlõ slave portjára.

   Most már begépelhetjük, hogy `boot`, és így a rendszernek el is kell indulnia.

   Ha ezt a változtatást véglegesíteni akarjuk (vagyis nem akarjuk ugyanezt eljátszani a FreeBSD minden egyes indítása során), akkor a /boot/loader.conf.local állományba vegyünk fel a `root_disk_unit="lemezszám"` sort.

2. Tegyük át a FreeBSD-t tartalmazó lemezt az elsõdleges IDE-vezérlõre, és ezzel megszûnik az iménti félreértés.

3.23. Mennyi memóriát tudunk használni?

A memóriára vonatkozó korlátozások platformonként változnak. Egy szabványos i386™ telepítés esetén például ez a határ 4 GB, de pae(4) segítségével akár még ennél több is elérhetõ. Ehhez olvassuk el az i386™ platformon 4 GB-nál több memória használatára vonatkozó utasításokat.

A FreeBSD/pc98 esetén a korlát szintén 4 GB, azonban itt a PAE nem használható. A FreeBSD által támogatott összes többi architektúra elméletileg ennél több memóriát képes kezelni (több terabyte-ot).

3.24. Mik az FFS állományrendszerek korlátai?

Az FFS állományrendszerek méretének elméleti határa 8 TB (2 milliárd blokk), illetve az alapértelmezett 8 KB-os blokkméret esetén 16 TB. A gyakorlatban azonban szoftveresen ebbõl 1 TB használható ki, de kisebb módosításokkal akár 4 TB-os állományrendszer is használható (és létezik).

Egyetlen FFS állományrendszerbeli állomány mérete megközelítõleg legfeljebb 1 milliárd blokk lehet, ami 4 KB-os blokkmérettel számolva 4 TB-ot jelent.

Táblázat 1. Az állományok maximális mérete

Blokkméret	Gyakorlatban	Elméletben
4 KB	> 4 GB	4 TB - 1
8 KB	> 32 GB	32 TB - 1
16 KB	> 128 GB	32 TB - 1
32 KB	> 512 GB	64 TB - 1
64 KB	> 2048 GB	128 TB - 1

4 KB-os blokkméret esetén a háromszoros indirekcióval származtatott blokkok a gyakorlatban is kihasználhatóak, és az egészet elméletben egyedül csak az állományrendszerben így ábrázolható blokkok maximális száma korlátozná (ami kb. 1024 + 1024 + 1024), azonban a gyakorlatban ezt az állományrendszeri blokkokra vonatkozó 1 GB - 1 méretû (rossz) határ korlátozza. Az állományrendszeri blokkok számát ugyanis ki kellene terjeszteni a 2 GB - 1 méretig. 2 GB - 1 számú blokk használata körül jelentkezik ugyan néhány hiba, de ezek 4 KB-os blokkméret esetén nem is érhetõek el.

A 8 KB-nál nagyobb blokkméretek esetén mindenre a blokkok 2 GB - 1 maximális mennyisége érvényes, de a gyakorlatban ezt a blokkok számának 1 GB - 1 határa korlátozza. Az eredeti 2 GB - 1 mennyiségû blokk használata gondokat okozhat.

3.25. Egy új rendszermag fordítása után miért jelenik meg a archsw.readin.failed hibaüzenet az indítás során?

Mert a rendszermag és a felhasználói programok verziója eltér. A rendszermag frissítésekor feltétlenül használjuk a `make buildworld` és a `make buildkernel` parancsokat is!

A rendszerindítás második fokozatában közvetlenül meg tudjuk adni a betöltendõ rendszermagot, ha a betöltõ indítása elõtt, a `|` jel megjelenésekor lenyomunk egy billentyût.

3.26. A telepítés megszakad a rendszer indítása közben, mit lehet ezzel kezdeni?

Próbáljuk meg letiltani az ACPI támogatást. Ezt úgy tudjuk megtenni, hogy amikor a rendszertöltõ elindul, lenyomjuk a `Szóköz` billentyût. Ekkor a következõt kapjuk:

```
OK
```

Itt gépeljük be az alábbi parancsot:

```
unset acpi_load
```

Majd ezt:

```
boot
```

Chapter 4. Hardverkompatibilitás

4.1. Általános kérdések

4.1.1. A FreeBSD rendszerükhöz szeretnénk hardvert vásárolni. Melyik gyártmány/márka/típus a legjobb?

Ez állandó téma a FreeBSD levelezési listákon. Mivel a hardverek gyorsan változnak, nem is számíthatunk másra. *Továbbra* is határozottan javasoljuk, hogy olvassuk át figyelmesen a FreeBSD 12.0 vagy 11.2 változatához tartozó hardverjegyzéket (Hardware Notes) és nézzünk után a levelezési listák archívumában mielõtt bármire is rákérdeznéünk a legfrissebb és legjobb hardverek ügyében. Könnyen elõfordulhat, hogy éppen a múlt héten esett szó arról a típusú eszközrõl, amirõl éppen érdeklõdni szeretnénk.

Ha laptoppal kapcsolatban lenne kérdésünk, akkor nézzük meg a FreeBSD laptop computer levelezési lista archívumát. Minden más esetben érdemes inkább a FreeBSD general questions levelezési lista archívumait megnézni vagy az adott hardverhez tartozó levelezési listát böngészni.

4.2. Memória

4.2.1. A FreeBSD képes 4 GB-nál, 16 GB-nál vagy akár 48 GB-nál több memóriát (RAM-ot) támogatni?

Igen. A FreeBSD operációs rendszerként képes az adott platformon kihasználni az összes rendelkezésre álló fizikai memóriát. Ne felejtsük el azonban, hogy az egyes platformokon ennek határa eltér. Például az i386™ platformon a PAE használata nélkül legfeljebb csak 4 GB memóriát tudunk elérni (amely azonban a PCI számára fenntartott címtér miatt a valóságban némileg kevesebb), illetve a PAE használatával legfeljebb 64 GB memóriát. Az AMD64 platformokon viszont már egészen 1 TB memóriáig is elmehetünk.

4.2.2. A FreeBSD miért jelez 4 GB-nál kevesebb memóriát i386™ architektúrájú számítógépeken?

Az i386™ platformon a címtér 32 bites, ami azt jelenti, hogy itt legfeljebb 4 GB memória címezhetõ meg (és érhetõ el). Ráadásul a címtér bizonyos tartományait a hardvereszközök számára tartják fenn különbözõ célokra, például a PCI eszközök mûködtetésére és vezérlésére, a videomemória hozzáférésére stb. Ennélfogva az operációs rendszer és annak rendszermagja által felhasználható teljes memória mérete jelentõsen kevesebb, mint 4 GB. Ezen a típusú konfigurációkon általában 3,2 GB és 3,7 GB között mozog a maximálisan kihasználható fizikai memória mérete.

Ha mégis 3,2 vagy 3,7 GB-nál több memóriát szeretnénk elérni (4 GB-ot vagy akár annál is többet), akkor ahhoz a PAE nevû speciális módosításra lesz szükségünk. A PAE a "Physical Address Extension" ("Fizikai címkiterjesztés") rövidítése, és egy olyan módszerre utal, amellyel a 32 bites x86 típusú processzorokon tudunk 4 GB-nál több memóriát címezni. Lényegében nem csinál mást, csak 4 GB-os határ felé képezi le azokat a memóriaterületeket, amelyeket egyébként a hardverek részére tartanak fenn, ezzel kiegészíti a fizikai memóriát (pae(4)). A PAE használatának számos

hátránya van: ebben a módban a megszokottnál (vagyis PAE nélkül) némileg lassabb a memória elérése, illetve ilyenkor a betölthetõ rendszermag-modulok (lásd kld(4)) sem támogatottak. Emiatt az összes meghajtót bele kell fordítanunk a rendszermagba.

A PAE használatát általában a PAE nevû, a rendszermaghoz gyárilag mellékelt konfigurációs állománnyal engedélyezhetjük. Ezt eleve úgy állították össze, hogy gond nélkül készíteni tudjuk egy ilyen rendszermagot. Érdemes azonban megemlíteni, hogy a konfigurációs állomány bizonyos tekintetben egy kissé konzervatív, mivel egyes PAE esetén használhatatlannak megjelölt meghajtók valójában mégis minden gond nélkül hozzáadhatóak a konfigurációhoz. Ezzel kapcsolatban azt javasoljuk, hogy ha az adott meghajtó használható valamelyik 64 bites architektúrán (például AMD64-en), akkor nagy valószínûséggel PAE-vel is mûködni fog. Amennyiben saját magunk szeretnénk egy PAE-rendszermagot készíteni, akkor a következõ sort tegyük bele a konfigurációs állományba:

```
options      PAE
```

A PAE alkalmazása napjainkban annyira már nem jellemzõ, mivel az újabb x86 hardverek mindegyike képes 64 bites (AMD64 vagy Intel® 64) módban futni. Ebben az esetben már lényegesen nagyobb címtér használatára nyílik lehetõségünk, így nincs szükségünk további trükkökre. A FreeBSD támogatja az AMD64 architektúrát, így ha 4 GB-nál több memóriát szeretnénk elérni, akkor inkább a FreeBSD ezen változatát érdemes alkalmazni.

4.3. Architektúrák és processzorok

4.3.1. A FreeBSD az x86-on kívül támogat más architektúrájú rendszereket is?

Igen. A FreeBSD jelenleg az Intel x86 és az AMD64 architektúrákon mûködik. A Az Intel EM64T, IA-64, ARM®, PowerPC®, sun4v és sparc64 architektúrák is támogatottak. A további tervezett platformok között van még a MIPS® és az S/390®, a MIPS® aktuális állapotáról és FreeBSD MIPS levelezési lista segítségével értesülhetünk. Az újabb architektúrákhoz kapcsolódó általános jellegû megbeszéléseket a FreeBSD non-Intel platforms levelezési lista foglalja össze.

Amennyiben a számítógépünk architektúrája nem szerepel a jelenleg támogatottak között, és valamilyen gyors megoldásra lenne szükségünk, akkor javasoljuk a NetBSD vagy az OpenBSD használatát.

4.3.2. A FreeBSD támogatja a szimmetrikus többprocesszoros (SMP) rendszereket?

A FreeBSD általánosságban véve támogatja a többprocesszoros rendszereket, noha egyes esetekben a BIOS vagy az alaplap hibájából fakadóan problémáink adódhatnak. A FreeBSD symmetric multiprocessing levelezési lista átolvasása segíthet tisztázni ezeket.

A FreeBSD képes kihasználni az Intel processzorai által felkínált HyperThreading (HTT) támogatás elõnyeit. Az `options SMP` beállítással fordított rendszermagok alapból maguktól felismerik a rendszerünkben található logikai processzorokat. A FreeBSD alapértelmezett ütemezõje ezeket a

logikai processzorokat a többivel teljesen egyenrangúnak tekinti, vagyis semmilyen ütemezési kérdés eldöntésénél nem fogja figyelembevenni az egy processzoron belül elhelyezkedõ logikai processzorokat. Ezen naív ütemezési felfogás miatt bizonyos esetekben a rendszerünk teljesítménye nem tökéletesen optimális, ezért adódhatnak olyan helyzetek, amikor a `machdep.hlt_logical_cpus` sysctl-változó segítségével szükséges lehet a logikai processzorok használatának letiltása. Ezenkívül még a `machdep.hlt_logical_cpus` sysctl-változón keresztül lehetõségünk van leállítani az üresjáratban mûködõ processzorokat. Ennek részleteirõl bõvebben a smp(4) man oldalon olvashatunk.

4.4. Merevlemezes, szalagos, CD- és DVD-meghajtók

4.4.1. A FreeBSD milyen típusú merevlemezes meghajtókat ismer?

A FreeBSD ismeri az EIDE-, SATA-, SCSI- és SAS-meghajtókat (és a velük kompatibilis vezérlõket, errõl bõvebben lásd a következõ szakaszt), valamint az összes olyan meghajtót, amely az eredeti "Western Digital" (MFM, RLL, ESDI és természetesen az IDE) interfészt használja. Néhány egyedi fejlesztésû ESDI vezérlõ nem fog mûködni, ezért lehetõleg maradjunk a WD1002/3/6/7 interfészeknél és azok másolatainál.

4.4.2. Milyen SCSI- vagy SAS-vezérlõket ismer?

A teljes listát a FreeBSD hardverjegyzékében találhatjuk meg a 12.0 vagy 11.2 kiadásban.

4.4.3. Milyen szalagos meghajtókat ismer?

A FreeBSD a SCSI és QIC-36 (QIC-02 interfésszel) szabványokat ismeri. Ezek közé értendõek a 8 mm-es (más néven Exabyte) és DAT-meghajtók is.

Bizonyos régebbi 8 mm-es meghajtók nem egészen kompatibilisek a SCSI-2 szabvánnyal, ezért a FreeBSD-vel sem feltétlenül képesek együttmûködni.

4.4.4. A FreeBSD támogatja a szalagok cseréjét?

A FreeBSD ch(4) eszközön és a chio(1) parancson keresztül támogatja a SCSI szabványú szalagcserélõket. A használat pontos részleteirõl a chio(1) man oldalán olvashatunk részletesebben.

Ha nem az AMANDA vagy a hozzá hasonló programokat használjuk, amelyek alapból ismerik a szalagcserélés lehetõségét, akkor ne feledkezzünk meg arról, hogy a szalagot csak az egyik helyrõl a másikra tudjuk mozgatni, ezért nekünk kell figyelnünk arra, hogy melyik rekeszben vannak szalagok és a meghajtónak ezek közül melyiket kell használnia.

4.4.5. A FreeBSD milyen CD-meghajtókat ismer?

Bármilyen támogatott SCSI-vezérlõhöz csatlakoztatható SCSI-meghajtót ismer.

Ezenkívül még az alábbi CD-interfészek ismertek:

- Mitsumi LU002 (8 bites), LU005 (16 bites) és FX001D (16 bites, dupla sebességû).

- Sony CDU 31/33A
- Sound Blaster nem-SCSI CD-meghajtók
- Matsushita/Panasonic CD-meghajtók
- ATAPI kompatibilis IDE CD-meghajtók

Az összes ismert nem-SCSI kártya nagyon lassan mûködik a SCSI-meghajtókhoz képest, és bizonyos ATAPI CD-meghajtók nem használhatóak.

A Daemon News-tól és a FreeBSD Mall-tól rendelhetõ hivatalos FreeBSD CD-krõl akár közvetlenül el is tudjuk indítani a rendszert.

4.4.6. A FreeBSD milyen CD-RW meghajtókat ismer?

A FreeBSD bármilyen ATAPI-kompatibilis IDE CD-R vagy CD-RW meghajtót ismer. Ennek részleteit lásd a burncd(8) man oldalán.

A FreeBSD ezeken kívül még tetszõleges SCSI CD-R vagy CD-RW meghajtót támogat. A használatukhoz telepítsük a `cdrecord` programot a portok vagy csomagok közül, és gondoskodjunk róla, hogy a pass eszköz támogatása benne legyen a rendszermagban.

4.4.7. A FreeBSD ismeri az Zip® meghajtókat?

A FreeBSD alapból ismeri a SCSI és ATAPI (IDE) interfészen kommunikáló Zip® meghajtókat. A SCSI ZIP-meghajtók ugyan egyedül az 5 és 6 target ID-krõl hajlandóak mûködni, de ha a SCSI-kártyánk BIOS-a támogatja, akkor még a rendszert is el tudjuk indítani róluk. Egyelõre nem tisztázott, hogy milyen kártyák képesek a 0 és 1 ID-ken kívül máshonnan is rendszert indítani, ezért ennek a hozzá tartozó dokumentációben érdemes utánajárnunk.

A FreeBSD ezenkívül még a párhuzamos porton csatlakoztatható ZIP-meghajtókat is ismeri. Ehhez ellenõrizzük, hogy a rendszermagunkban megtalálhatóak az scbus0, da0, ppbus0 és vp0 meghajtók (a GENERIC rendszermagban a vp0 kivételével mindegyik szerepel). Segítségükkel a párhuzamos vonalon csatlakozó meghajtó a da0s4 eszközön keresztül érhetõ el. Ennek megfelelõen az állományrendszerek a `mount /dev/da0s4 /mnt` *vagy* (DOS esetén) a `mount -t msdosfs /dev/da0s4 /mnt` parancs kiadásával csatlakoztathatóak.

Emellett még érdemes a GYIK cserélhetõ lemezes meghajtókról szóló részét is elolvasnunk ebben a fejezetben, valamint a "formázásról" szóló megjegyzést az adminisztrációról szóló fejezetben.

4.4.8. A FreeBSD ismeri a Jaz®, EZ és a többi cserélhetõ lemezes meghajtót?

Használhatóak. Ezek többsége SCSI eszköz, ezért a FreeBSD SCSI-lemezként látja, az IDE csatolós EZ pedig IDE-meghajtóként érhetõ el.

A rendszer indítása elõtt ne felejtsük el bekapcsolni a külsõ egységeket.

Ha tárolóeszközt akarunk cserélni a rendszer mûködése közben, olvassuk el a mount(8), umount(8) és (SCSI eszközök esetén) a camcontrol(8) vagy (IDE eszközök esetén) a atacontrol(8) man oldalakat, valamint a GYIK egy késõbbi részében található részt a cserélhetõ lemezes meghajtókról.

4.5. Egér és billentyûzet

4.5.1. A FreeBSD ismeri az USB billentyûzeket?

A FreeBSD alapból ismeri az USB billentyûzeket. Miután engedélyeztük rendszerünkben az USB billentyûzet támogatását, az AT billentyûzet /dev/kbd0 lesz és az USB billentyûzet pedig /dev/kbd1, már amennyiben mind a kettõt csatlakoztattuk a számítógépünkhöz. Ha viszont csak USB billentyûzetünk van, akkor az a /dev/ukbd0 lesz.

Ha az USB billentyûzetet konzolban akarjuk használni, akkor erre figyelmeztetnünk kell a konzolos meghajtót. Ezt úgy tudjuk megtenni, ha a következõ parancsot lefuttatjuk a rendszer indítása közben:

```
# kbdcontrol -k /dev/kbd1 < /dev/console > /dev/null
```

Amikor viszont csak USB billentyûzetünk van, akkor az /dev/ukbd0 eszközön keresztül tudjuk elérni, ezért a parancsnak ilyenkor így kell kinéznie:

```
# kbdcontrol -k /dev/ukbd0 < /dev/console > /dev/null
```

Ha véglegesíteni akarjuk ezt a beállítást, akkor tegyük a `keyboard="/dev/ukbd0"` sort az /etc/rc.conf állományba.

Miután ezt megcsináltuk, az USB billentyûzet X alatt is mûködni fog minden további beállítás nélkül.

Ezzel a paranccsal tudunk visszaváltani az alapértelmezett billentyûzetre:

```
# kbdcontrol -k /dev/kbd0 > /dev/null
```

A kbdmux(4) meghajtón keresztül az alábbi parancsok kiadásával engedélyezhetjük az elsõdleges AT billentyûzet és a másodlagos USB billentyûzet párhuzamos használatát a konzolon:

```
# kbdcontrol -K < /dev/console > /dev/null
# kbdcontrol -a atkbd0 < /dev/kbdmux0 > /dev/null
# kbdcontrol -a ukbd1 < /dev/kbdmux0 > /dev/null
# kbdcontrol -k /dev/kbdmux0 < /dev/console > /dev/null
```

Részletesebb információkat az ukbd:(4), kbdcontrol(1) és kbdmux(4) man oldalakon találhatunk.

Az USB billentyûzet menet közbeni csatlakoztatása és leválasztása nem feltétlenül fog mûködni. Ezért a problémák elkerülése érdekében azt javasoljuk, hogy a rendszer indítása elõtt mindenképpen csatlakoztassuk a billentyûzetet és hagyjuk egészen úgy, amíg le nem állítottuk.

4.5.2. A nem szabványos buszos egereket hogyan lehet beállítani?

A FreeBSD ismeri a buszos, illetve a Microsoft, Logitech és az ATI által gyártott InPort buszos egereket. A GENERIC rendszermag azonban ehhez nem tartalmaz meghajtót. A rendszermag konfigurációs állományába a következõ sort kell megadni, ha egy buszos egereket támogató rendszermagot akarunk készíteni:

```
device mse0 at isa? port 0x23c irq5
```

A buszos egerekhez általában saját interfészkártya is tartozik. Ezeket a kártyákat a fentitõl eltérõ portcímre és IRQ megszakításra is beállíthatjuk. Részletesebb információkat az egerünk man oldalán és a mse(4) man oldalon olvashatunk.

4.5.3. Hogyan lehet PS/2 (egérportos vagy billentyûzetes) egeret használni?

Az PS/2 egereket alapból támogatjuk. Az ehhez szükséges psm meghajtó megtalálható a rendszermagban.

Ha a saját magunk által összeállított rendszermagunk nem tartalmazza ezt a meghajtót, akkor a következõ sort kell felvennünk a konfigurációs állományba:

```
device psm0 at atkbdc? irq 12
```

Miután a rendszermag a rendszer indítása során helyesen észlelte a psm0 eszközt, magától létrejön.

4.5.4. Az egeret az X Window Systemen kívül is lehet valamilyen módon használni?

Ha az alapértelmezett konzolos syscons(4) meghajtót használjuk, akkor a szöveges felületû konzolokon az egérmutató segítségével tudunk szövegrészeket kijelölni és másolni. Ehhez nem kell mást tennünk, csupán elindítani a moused(8) egérdémont és engedélyezni az egérmutatót a virtuális konzolokon:

```
# moused -p /dev/xxxx -t yyyy
# vidcontrol -m on
```

Itt az *xxxx* az egeret leképezõ eszköz neve és az *yyyy* az egérhez használt protokoll típusa. Az egérdémon a legtöbb egér esetén képes magától megállapítani az alkalmazott protokoll típusát, kivéve a régebbi soros egereket. Az `auto` érték megadásával tudjuk aktiválni ezt az automatikus felderítést. Amennyiben ez nem mûködik, a moused(8) man oldalán nézhetünk után a támogatott protokolloknak.

Ha PS/2 egerünk van, akkor egyszerûen csak vegyük fel a `moused_enable="YES"` sor az /etc/rc.conf állományba, és az egérdémon elindul a rendszer indítása közben. Valamint hogy ha az egérdémont a konzol helyett az összes virtuális konzolon is használni akarjuk, akkor az /etc/rc.conf állományba tegyük bele a `allscreens_flags="-m on"` sort.

Miután az egérdémon elindult, valamilyen módon koordinálni kell az egér hozzáférését az egérdémon és az összes többi program, például az X Window System között. Errõl a problémáról a GYIK Miért nem mûködik X alatt az egér? kérdésében olvashatunk részletesebb.

4.5.5. Hogyan lehet szöveget kijelölni és másolni a szöveges konzolban?

Ahogy sikerült elindítanunk az egérdémont (lásd az elõzõ szakaszt), tartsuk lenyomva az egér elsõ (bal oldali) gombját és az egér mozgatásával jelöljük ki a szöveget. Ezután nyomjuk le a második (középsõ) gombját, amivel a kurzor mellett megjelenik az imént kijelölt szöveg. A harmadik (jobb oldali) gomb segítségével a szöveg kijelölését tudjuk "kiterjeszteni".

Amennyiben az egerünkön nem található középsõ gomb, az egérdémon beállításainak segítségével megpróbálkozhatunk emulálni vagy áthelyezni a vele kapcsolatos funkciókat egy másik gombra. A moused(8) man oldalán olvashatunk errõl részletesebben.

4.5.6. Az egéren van mindenféle görgõ és gomb. Ki lehet ezeket valahogy használni FreeBSD alatt is?

A válaszunk erre sajnos csupán annyi, hogy "Attól függ". A különbözõ kiegészítõkkel rendelkezõ egerekhez általában egy külön meghajtó szükségeltetik. Hacsak az egér meghajtóprogramja vagy a hozzá tartozó felhasználói program nem nyújt valamilyen támogatást, az eszköz egyszerûen csak egy szabványos két- vagy háromgombos egérként fog funkcionálni.

Ha az X Window környezetben akarunk görgõket használni, esetleg ezt a szakaszt érdemes elolvasnunk.

4.5.7. A laptopokon megtalálható egér/trackball/touchpad hogyan használható?

Olvassuk el az elõzõ kérdésre adott választ.

4.5.8. A Delete billentyû hogyan használható a sh és csh parancsértelmezõkben?

A Bourne Shell esetében az alábbi sorokat kell megadnunk az .shrc állományunkban. Lásd sh(1) és editrc(5).

```
bind ^? ed-delete-next-char # a konzolhoz
bind ^[[3~ ed-delete-next-char # az xtermhez
```

A C Shell esetében a következõ soroknak kell az .cshrc állományba kerülnie. Lásd csh(1).

```
bindkey ^? delete-char # a konzolhoz
bindkey ^[[3~ delete-char # az xtermhez
```

További információkat ezen az oldalon találhatunk.

4.6. Hálózati és soros eszközök

4.6.1. A FreeBSD milyen hálózati kártyákat ismer?

Ezek teljes listáját a FreeBSD egyes kiadásaihoz tartozó hardverjegyzékben találjuk meg.

4.6.2. A FreeBSD ismer szoftveres modemeket, például winmodemeket?

A FreeBSD különbözõ kiegészítõ szoftvereken keresztül több szoftveres modemet is támogat. A comms/ltmdm port például a szélesebb körben elterjedt Lucent LT chipsetes modemekhez ad támogatást.

A FreeBSD azonban nem telepíthetõ szoftveres modemen keresztül. A hozzá tartozó szoftvert csak az operációs rendszer telepítése után tudjuk telepíteni.

4.6.3. Van natív meghajtó a Broadcom 43xx típusú kártyákhoz?

Nem, és valószínûleg nem is lesz.

A Broadcom nem hajlandó nyilvánossá tenni azokat az információkat, amik az általuk gyártott vezeték nélküli chipsetek programozásához lennének szükségesek, mivel szoftveresen vezérelt rádiót használnak. Az alkatrészeik FCC szintû engedélyeztetéséhez ugyanis valamilyen módon gondoskodniuk kell róla, hogy a felhasználók nem képesek bizonyos dolgokat módosítani vele kapcsolatban, például a mûködési frekvenciát, a modulációs paramétereket vagy a kimenõ teljesítményt. A chipsetek programozásának ismerete nélkül azonban szinte lehetetlen elkészíteni hozzájuk a megfelelõ meghajtót.

4.6.4. A FreeBSD milyen többportos soros vonali kártyákat ismer?

Ezek listáját a kézikönyv Soros vonali kommunikációról szóló része tartalmazza.

Bizonyos névtelen másolatok is használhatók, különösen azok, amelyek magukat AST-kompatibilisnek nevezik.

Az ilyen kártyák beállításáról a sio(4) man oldalon olvashatunk részletesebben.

4.6.5. Hogyan lehet a boot: parancssort elõhozni soros vonali konzolon?

Olvassuk el a kézikönyvben ezt a fejezetet.

4.7. Hang

4.7.1. A FreeBSD milyen hangkártyákat ismer?

A FreeBSD rengeteg hangkártyát ismer, (ennek részleteit lásd a FreeBSD kiadásait tartalmazó honlapon és a snd(4) man oldalon). Korlátozott módon az MPU-401 és a vele kompatibilis MIDI-kártyákat is támogatja. A Microsoft® Sound System specifikációinak megfelelõ kártyákat tudjuk használni.

Ez azonban csak a hangra vonatkozik! Ez a meghajtó a SoundBlaster® kivételével nem támogatja a kártyákon található CD-, SCSI- és joystick csatlakozásokat. A SoundBlaster® SCSI csatlakozása és bizonyos nem-SCSI CD-meghajtókat ugyan támogat, de rendszert például nem tudunk róluk indítani.

4.7.2. Miért nincs hang a pcm(4) által támogatott hangkártyán?

Egyes hangkártyák esetében a hangerõ minden indításkor nullára állítódik. Ezért ilyenkor mindig ki kell adni a következõ parancsot:

```
# mixer pcm 100 vol 100 cd 100
```

4.8. Egyéb eszközök

4.8.1. Képes a FreeBSD kihasználni az energiagazdálkodási lehetõségeket egy laptopon?

A FreeBSD bizonyos gépeken képes az APM használatára. Errõl az apm(4) man oldalon találunk pontosabb leírást.

A FreeBSD ezenkívül még a legújabb hardverekben megtalálható ACPI lehetõségeit is igyekezik kihasználni. Errõl részletesebben az acpi(4) man oldalon olvashatunk. Amennyiben a rendszerünk egyaránt tartalmazza az APM és az ACPI támogatását, bármelyiket használhatjuk. Ilyen esetben javasoljuk mind a kettõ kipróbálását és az igényeinkhez leginkább illeszkedõ megoldás kiválasztását.

4.8.2. Hogy lehet letiltani az ACPI támogatását?

Tegyük bele az alábbi sort az /boot/device.hints állományba:

```
hint.acpi.0.disabled="1"
```

4.8.3. Miért fagynak le a Micron típusú rendszerek indulás közben?

Egyes Micron gyártmányú alaplapokon olyan PCI BIOS található, amely nem felel meg az szabványoknak, és ezért a FreeBSD nem tud elindulni, mivel a PCI eszközök nem jelentik le az általuk használt címeket.

Ezt a problémát úgy tudjuk megoldani, ha a BIOS-ban kikapcsoljuk (`Disabled` értékûre állítjuk) a "Plug and Play Operating System" beállítást.

4.8.4. A rendszerindító lemez nem képes az ASUS K7V alaplapokkal mûködni. Hogyan lehet ezt orvosolni?

Menjünk be a BIOS-ba és kapcsoljuk ki (állítsuk `Disabled` értékre) a "Boot Virus Protection"

beállítást.

4.8.5. Miért nem mûködnek a 3Com® PCI hálózati kártyák a Micron típusú számítógépekben?

Nézzük meg az elõzõ választ.

Chapter 5. Hibaelhárítás

5.1. Miért állapítja meg rosszul a FreeBSD a memória mennyiségét i386™ hardveren?

A válasz nagy valószínûséggel a fizikai és virtuális memóriacímek közti különbségben rejlik.

A legtöbb PC-s hardvereszköz megegyezés szerint a 3,5 GB és 4 GB közti memóriaterületet speciális célokra tartja fenn (általában a PCI számára). Ezen a címterületen keresztül éri a PCI eszközöket. Ennek egyik következménye, hogy a fizikai memória ezen a részen nem érhetõ el.

Hogy pontosan mi történik az itt elhelyezkedõ memóriával, teljesen a hardvertõl függ. Sajnálatos módon bizonyos eszközök semmilyen megoldást nem nyújtanak a problémára, és így lényegében az utolsó 500 MB-nyi memória elveszik.

Szerencsére a legtöbb eszköz azonban képes ezt a területet egy felsõbb címre leképezni, így ki tudjuk használni. Ilyenkor azonban tapasztalhatunk némi félreértést, amikor megnézzük a rendszerindítás közben megjelenõ üzeneteket.

A FreeBSD 32 bites változata esetén ez a memóriaterület elveszik, mivel a címe a 4 GB-os határ felé kerül, amelyet a 32 bites módban futó rendszermag már nem képes elérni. Ezen egy PAE támogatással rendelkezõ rendszermag használatával segíthetünk. A GYIK-on belül ebben a bejegyzésben olvashatunk bõvebben a memóriakorlátokról, valamint ebben a részben láthatjuk a különbözõ platformokra vonatkozó memóriakorlátozásokat.

A FreeBSD 64 bites változata vagy a PAE használata esetén azonban a FreeBSD rendesen felismeri és leképezi a fennmaradó memóriaterületeket, így azok használhatóvá válnak. A rendszerindítás során azonban az elõbb említett leképezés miatt látszólag úgy fog tûnni, mintha a FreeBSD több memóriát észlelne, mint amennyivel valójában rendelkezünk. Ez teljesen normálisnak tekinthetõ és a ténylegesen elérhetõ memória mennyisége a folyamat végén be fog állítódni.

5.2. Mit tegyünk, ha meghibásodott szektorokat találunk a merevlemezünkön?

A SCSI-meghajtók esetében a meghajtó általában képes önmagától átképezni az ilyen szektorokat. A legtöbb meghajtóban ez a lehetõség viszont alapból nem engedélyezett.

A hibás szektorok átképezéséhez az eszköz elsõ lapmódját kell átírnunk, amelyet (`root` felhasználóként) így tehetünk meg:

```
# camcontrol modepage sd0 -m 1 -e -P 3
```

Változtassuk meg az AWRE (az írás automatikus átképzése) és ARRE (az olvasás automatikus átképzése) beállítások értékeit 0-ról 1-re:

```
AWRE (Auto Write Reallocation Enbld):  1
ARRE (Auto Read Reallocation Enbld):  1
```

A modernebb IDE-meghajtók is képesek a vezérlõjükkel nyilvántartani az idõközben meghibásodott szektorokat, és ezt általában alapból engdélyezik.

Ha rossz szektorokra figyelmeztetõ hibaüzeneteket látunk (akármilyen típusú meghajtónk is legyen), az kétségtelenül arra utal, hogy ideje lecserélnünk a hardvert. A hibás szektorok használatát esetleg a gyártó saját diagnosztikai programjával le tudjuk tiltani, de hosszabb távon mindenképpen az lesz a legjobb, ha veszünk egy újat.

5.3. A FreeBSD miért nem találja meg a HP Netserver SCSI-vezérlõjét?

Ez tulajdonképpen egy ismert probléma. A HP Netserver gépekben egy integrált EISA buszos SCSI-vezérlõ található, amely a 11-es EISA bõvítõhelyen található, ezért az összes "valódi" EISA bõvítõhely ez elõtt helyezkedik el. Sajnos a 10 feletti EISA bõvítõhelyek címei ütköznek a PCI eszközök számára kiosztott címekkel, ezért a FreeBSD önmagától nem tudja valami jól kezelni az ilyen helyzeteket.

Ezért a legjobban akkor járunk, ha egyszerûen letagadjuk a címterek ütközését :) Ezt úgy tudjuk megtenni, ha a rendszermag `EISA_SLOTS` nevû beállítását a 12 értékre állítjuk. Ezután már csak be kell konfigurálunk és újra kell fordítanunk a rendszermagot, ahogy azt a kézikönyv megfelelõ része is tárgyalja.

Természetesen, amikor egy ilyen gépre akarunk telepíteni, a helyzet tovább bonyolódik. A telepítést úgy tudjuk megoldani, ha a *UserConfig* programon belül alkalmazunk egy apró trükköt. Most ne a "vizuális" felületét használjuk, hanem a parancssoros részt. Gépeljük be, majd a megszokottak szerint telepítsük a rendszert:

```
eisa 12
quit
```

Ettõl függetlenül természetesen továbbra is javasolt egy, az elõbbiek szerint módosított rendszermagot fordítanunk és telepítenünk.

A következõ verziókban remélhetõleg már lesz valamilyen megoldás erre a problémára.

A HP Netserver esetén nem tudunk a lemezeken `Veszélyesen dedikált` (`Dangerously Dedicated`) módot használni. Errõl itt olvashatunk bõvebben.

5.4. Állandóan ed1: timeout és ahhoz hasonló üzenetek jelennek meg. Mi lehet velük kezdeni?

Ezt a hibát általában a megszakítások ütközése okozza (például két kártya ugyanazt a megszakítást

akarja használni). Indítsuk a rendszerünket a `-c` beállítás használatával és az ed0/de0/... bejegyzéseket változtassuk meg a kártyáknak megfelelõen.

Ha a hálózati kártyánkon BNC típusú csatlakozó található, akkor még elõfordulhat, hogy azért látunk ilyen hibaüzeneteket, mert nem jól zártuk le a csatlakozást. Ezt úgy tudjuk könnyen ellenõrizni, ha a lezárót közvetlenül a kártyára dugjuk rá (kábel nélkül) és figyeljük, hogy továbbra is jönnek-e a hibaüzenetek.

Egyes NE2000-kompatibilis kártyák akkor adják ezt a hibát, ha az UTP portjukon nincs aktív összeköttetés vagy nem dugtuk be a kábelt.

5.5. Miért állnak le a 3Com® 3C509 kártyák minden különösebb ok nélkül?

Az ilyen típusú kártyák néha hajlamosak elfelejteni a beállításaikat. Frissítsük a kártya beállításait a `3c5x9.exe` program segítségével.

5.6. A párhuzamos nyomtató nevetségesen lassú. Mi lehet ezzel kezdeni?

Ha csupán annyi a problémánk, hogy a nyomtató irdatlanul lassan mûködik, akkor próbáljuk meg a kézikönyv nyomtatásról szóló részében leírtakhoz hasonlóan átállítani a nyomtató portkezelését.

5.7. A programok miért állnak le idõnként Signal 11 hibákkal?

Ezek a hibák akkor keletkeznek, amikor a futó programok olyan memóriaterülethez próbálnak meg hozzáférni, amihez eredetileg nem lenne szabad. Ha valami ehhez hasonló történik a rendszerünkben látszólag teljesen véletlenszerûen, akkor nagyon óvatosan kezdjünk el vizsgálódni.

A lehetséges okok az alábbiak lehetnek:

1. Ha csak olyan alkalmazások esetében jelentkezik ez a hiba, amelyeket mi magunk fejlesztünk, akkor az valószínûleg arra utal, hogy valamelyik része hibásan mûködik.
2. Ha a FreeBSD alaprendszerének valamelyik részében tapasztalunk ilyen hibákat, akkor azt szintén okozhatja hibás kód, de az ilyen hibákat általában hamarabb meg szokták találni és ki szokták javítani, mint ahogy a GYIK-ot olvasók többsége találkozna velük (a `-CURRENT` ág pontosan ezt a célt szolgálja).

Elõfordulhat, hogy ez egy olyan furcsaság eredménye, amely *nem* a FreeBSD hibája: például ugyanazon program fordításakor mindig mást csinál a fordítóprogram.

Például tegyük fel, hogy a `make buildworld` parancsot futtatjuk, és a fordítás félbeszakad, amikor az ls.c állományból el akarja készíteni az ls.o állományt. Ha ezután megint megpróbáljuk kiadni a `make buildworld` parancsot, akkor a fordítás ugyanazon a helyen újból meghiúsul - valószínûleg hibás a forráskód, frissítsük a forrásainkat és próbáljuk meg ismét. Ha viszont a fordítás ilyenkor már egy

másik helyen akad el, akkor szinte biztos, hogy hardverhibával akadtunk össze.

Amit ilyenkor tenni tudunk:

Az elsõ esetben egy nyomkövetõ, például a gdb(1) segítségével keressük meg a program azon pontját, ahol rossz memóriaterülethez próbál meg hozzáférni és javítsuk ki.

A második esetben ellenõrizzük, hogy nem a hardver a hibás.

Ennek okai többek közt a következõk lehetnek:

1. Túlmelegednek a merevlemezeink: ellenõrizzük, hogy a gépben található ventillátorok rendesen mûködnek-e (persze elõfordulhat, hogy más eszközök melegednek túl).
2. A processzor túlmelegedett: lehet, hogy mert túlságosan nagy órajelen járatjuk, vagy mert egyszerûen leállt a hûtése. Akármelyik eset is következett be, legalább a hiba felderítéséig állítsuk vissza a hivatalos sebességére.

   Ha feltétlenül ragaszkodunk a rendszerünk tuningolásához, akkor érdemes elgondolkoznunk azon, hogy egy lassabb rendszerrel jobban járunk, mint egy állandóan cserélendõ, ropogósra sült rendszerrel. Az emberek általában nem is nagyon szeretik az ilyen rendszereket, független attól, hogy szerintünk érdemes-e ilyet csinálni vagy sem.

3. Hibás memóriamodulok: ha több SIMM és DIMM modul is található a gépünkben, akkor vegyük ki az összeset és próbáljuk ki mindegyiket egyesével, ezzel is leszûkíthetjük a probléma felderítését a hibás DIMM/SIMM modulokra vagy azok kombinációjára.
4. Az alaplap túlbecslõ értékei: a BIOS beállításai között vagy az alaplapon található jumperekkel szabályozni tudjuk a különbözõ idõzítéseket, ahol általában az alapértelmezett értékek megfelelnek, de néha elõfordulhat, hogy a memóriamodulok késleltetését lassúra, vagy éppen turbó sebességre állítják ("RAM Speed: Turbo" vagy ehhez hasonló néven keressük a BIOS-ban), ami szintén okozhat furcsa viselkedést. Próbáljuk meg visszaállítani az BIOS alapértelmezett értékeit, de elõtte érdemes lejegyezni az aktuális beállításainkat.
5. Az alaplap zajos vagy kevés áramot kap: ha vannak használaton kívüli I/O kártyáink, merevlemezeink, CD-meghajtóink a rendszerünkben, akkor próbáljuk meg ideiglenesen eltávolítani ezeket vagy egyszerûen csak lehúzni róluk a tápkábelt. Ezzel tudjuk vizsgálni, hogy a számítógépünk tápegysége képes-e megbirkózni a kisebb terheléssel. Esetleg kipróbálhatunk egy másik tápegységet is, lehetõleg egy kicsivel erõsebbet (például ha a jelenlegi tápegységünk teljesítménye 250 watt, akkor használjunk helyette egy 300 wattosat).

Továbbá érdemes lehet még elolvasnunk a SIG11 GYIK-ot (lásd lentebb), ahol mindezeket a problémákat részletesen kifejtik, noha a Linux® nézõpontjából. Arról is olvashatunk benne, hogy egy hibás memóriát miért nem képesek észlelni a szoftveres vagy hardveres tesztelõeszközök.

Végezetül, ha az egyik javaslat sem segített a probléma megoldásában, akkor valószínûleg sikerült hibát találnunk a FreeBSD kódjában, amirõl nyugodtan írhatunk a fejlesztõknek egy hibajelentést.

A problémáról minden részletre kiterjedõ módon A SIG11-es probléma GYIK-ja írásban olvashatunk (angolul).

5.8. A rendszer összeomlik vagy egy Fatal trap 12: page fault in kernel mode vagy pedig valamilyen panic: hibaüzenettel és egy halom számot ír ki. Mit tegyünk?

A FreeBSD fejlesztõi nagyon kíváncsiak az ilyen hibákra, de a felderítéséhez sajnos jóval több információra van szükségük, mint amennyit láthattunk. Másoljuk le az összeomláshoz tartozó teljes üzenetet. Ezután nézzük meg a GYIK-nak azt a részét, amely a rendszermag összeomlásáról szól, készítsünk egy nyomkövetési információkkal ellátott rendszermagot és kérjük le a hívási láncot. Ez elsõre talán bonyolultnak hangzik, de ehhez igazából nem igényel semmilyen programozási tudást, egyszerûen csak a megadott utasításokat kell követnünk.

5.9. A rendszer indulása közben miért sötétül a képernyõ és megy el rajta a kép?

Ez az ATI Mach 64 videokártyák esetében jelentkezõ probléma. Ilyenkor az a gond, hogy a kártya a `0x2e8` címet használja, akárcsak a negyedik soros port. A sio(4) meghajtóban levõ hiba (vagy netalán beállítás?) miatt azonban a negyedik soros portot *még* akkor is használni fogja, ha kikapcsoljuk a sio3 (a negyedik soros port) eszközt.

A hibát kijavításáig így kerülhetjük meg:

1. A betöltõ parancssorában adjuk meg a `-c` paramétert. (Így elõ tudjuk hozni a rendszermag konfigurációs módját.)
2. Kapcsoljuk ki a sio0, sio1, sio2 és sio3 eszközöket (tehát mindegyiket). Emiatt a sio(4) meghajtó nem indul el, és így nem okoz problémát.
3. Lépjünk ki és folytassuk a rendszer indítását.

Ha a soros portokat is használni akarjuk, akkor következõ módosításokkal készítsünk egy új rendszermagot: a /usr/src/sys/dev/sio/sio.c (vagy pc98 esetén a /usr/src/sys/pc98/cbus/sio.c) állományban keressük meg a `0x2e8` karakterláncot és az azt megelõzõ vesszõt távolítsuk el (de az utána következõt tartsuk meg). Miután végrehajtottuk ezt a módosítást, a megszokott módon fordítsuk újra a rendszermagot.

5.10. A FreeBSD miért csak 64 MB memóriát használ, amikor 128 MB van a gépben?

Mivel FreeBSD a BIOS-tól próbálja megtudni a rendelkezésre álló memória méretét, ezért csak 16 biten képes lekérdezni a KB-okban (vagyis 65 535 KByte = 64 MB, vagy még ennél is kevesebb, mivel egyes BIOS-ok legfeljebb 16 MB memóriát engednek látni). Tehát ha 64 MB-nál több memóriával rendelkezünk, akkor a FreeBSD ugyan megpróbálja azt felderíteni, de nem feltétlenül fog sikerülni.

Ezt úgy tudjuk megoldani, ha a rendszermag alábbi beállítását használjuk. Alapvetõen ugyanis létezik egy módszer, amivel le lehet kérdezni a memória teljes méretét a BIOS-tól, de a hozzá tartozó rutin nem fért el a rendszerindító blokkban. Ha egyszer majd sikerül neki helyet csinálni, akkor a rendszer képes lesz kizárólag ezzel a módszerrel dolgozni. Amíg viszont ez nem így van,

addig kénytelenek leszünk a most következõ megoldást választani:

```
options MAXMEM=N
```

ahol *N* a memória Kilobyte-okban megadott mérete. Tehát egy 128 MB memóriával rendelkezõ számítógép esetén ez 131072.

5.11. A számítógépben több mint 1 GB memória van, de mégis kmem_map too small üzenetek jelennek meg. Mi a gond?

A FreeBSD általában a rendszermag néhány fontos paraméterét, mint például az egyszerre megnyitható állományok maximális számát a számítógépben található memória méretébõl származtatja. Az 1 GB memóriánál több esetén azonban elképzelhetõ, hogy ez az "automatikus méretezés" túlságosan is nagy értékeket választ. Így a rendszer indításakor a rendszermag olyan nagy méretû táblázatokat és egyéb struktúrákat foglal le, amelyek betöltik a rendelkezésére bocsátott terület nagy részét. Késõbb, a rendszer futása közben pedig a rendszermag szépen lassan kifogy a dinamikus memóriaterületekbõl és összeomlik.

Készítsünk egy olyan saját rendszermagot, ahol a VM_KMEM_SIZE_MAX beállítást megnöveljük egészen a maximális 400 MB-os értékig (options VM_KMEM_SIZE_MAX=419430400). 400 MB használata valószínûleg elég lesz egészen 6 GB memóriáig.

5.12. A számítógépben nincs 1 GB memória, a FreeBSD mégis kmem_map too small hibával leáll!

Ez a hibaüzenet arra utal, hogy a rendszer kifogyott a hálózati pufferek (különösen az mbuf klaszterek) számára kiosztott virtuális memóriából. Az mbuf klaszterek részére fenntartott virtuális memória méretének beállításáról a kézikönyv Hálózati korlátozások címû szakaszában olvashatunk.

5.13. Miért jelenik meg a kernel: proc: table is full hibaüzenet?

A FreeBSD rendszermagja egyszerre csak bizonyos számú programot enged futni. Ezek konkrét száma a kern.maxusers sysctl(8)-változótól függ. A kern.maxusers ezenkívül még hatással van más belsõ korlátokra is, például a hálózati pufferekre (lásd ezt a korábbi kérdést). Ha a számítógépünk túlságosan leterhelt, akkor érdemes megpróbálkoznunk a kern.maxusers értékének növelésével. Ennek átállítása a rendszerben egyszerre futtatható maximális programok számával együtt sok más rendszerszintû korlátozást is finomít.

A kern.maxusers értékének beállításához nézzük meg a kézikönyv Az állományok és futó programok korlátozásairól szóló szakaszát. (Miközben ez a rész a megnyitható állományok maximális számáról szól, addig ugyanez érvényes a futó programokra is.)

Ha viszont a számítógépünk nem éri akkora terhelés, de mégis szeretnénk egyszerre nagyobb számú programot is futtatni rajta, akkor ehhez elegendõ csak `kern.maxproc` változót átállítanunk. Ezt úgy tudjuk megtenni, ha felvesszük a /boot/loader.conf állományba. Ez az érték természetesen addig nem beállítódni, amíg a rendszerünket újra nem indítjuk. Ezekrõl a változókról a loader.conf(5) és sysctl.conf(5) man oldalakon tájékozódhatunk részletesebben. Ha az összes programot egyetlen felhasználóval akarjuk futtatni, akkor a `kern.maxprocperuid` változót értékét is át kell állítanunk, méghozzá a `kern.maxproc` új értékénél eggyel kisebbre. (Ezért kell így csinálni, mert egy rendszerprogram, az init(8) mindig fut.)

A sysctl változók beállításait úgy is tudjuk véglegesíteni, ha felvesszük ezeket az /etc/sysctl.conf állományba. A kézikönyv A rendszermag korlátainak finomhangolása címû szakaszában részletesebb is olvashatunk róla, hogy miként állítsuk be a rendszerünket.

5.14. Az új rendszermag indításakor miért keletkezik CMAP busy hibaüzenet?

Az elavult /var/db/kvm_*.db állományokat összegyûjtõ rutin idõnként nem mûködik megfelelõen, és a nem egyezõ állományok esetén össze is omolhat.

Amikor ilyen történik, indítsuk újra a rendszert egyfelhasználós módban és gépeljük be:

```
# rm /var/db/kvm_*.db
```

5.15. Mit jelent az ahc0: brkadrint, Illegal Host Access at seqaddr 0x0 üzenet?

Ez az Ultrastor SCSI vezérlõkártya ütközésére utal.

A rendszerindítás közben lépjünk be a rendszermag konfigurációs menüjébe és tiltsuk le a gondot okozó uha0 eszközt.

5.16. Amikor elindul a rendszer, egy ahc0: illegal cable configuration hibaüzenet jelenik meg. A kábelek bekötésével semmilyen gond nincs. Mégis akkor mi a baj?

Az alaplapon nem található olyan áramkör, amely támogatja az automatikus lezárást ("automatic termination"). A SCSI BIOS-ban az automatikus lezárás helyett adjuk meg a megfelelõ lezárást. Az ahc(4) meghajtója nem képes rendesen érzékelni a kábeleket, ha az alaplapon van ilyen érzékelés (és így automatikus lezárás). A meghajtó egyszerûen annyit feltételez, hogy ennek támogatása csak akkor érhetõ el, ha az EEPROM-ban megadtuk az "automatic termination" beállítást. A megfelelõ kábeldetektáló eszköz nélkül a meghajtó gyakran rosszul állapítja meg a lezárást, ami pedig így veszélyezteti a SCSI busz megbízhatóságát.

5.17. Miért küld a sendmail mail loops back to myself hibaüzenetet?

Errõl részletesebben a kézikönyvben olvashatunk.

5.18. A távoli gépeken miért viselkednek olyan furcsán a teljes képernyõs alkalmazások?

Elõfordulhat, hogy az adott távoli gépen a terminál típusa nem `cons25`, amire viszont a FreeBSD konzolnak a megfelelõ mûködéshez szüksége lenne.

Ezt a problémát többféle módon is meg tudjuk kerülni:

- Mikor bejelentkezünk a távoli gépre, állítsuk a `TERM` környezeti változót az `ansi` vagy `sco` értékre, amibõl kiderül, hogy egyáltalán ismeri ezeket a terminéltípusokat.
- A FreeBSD konzolban használjunk VT100 emulátort, például a screen alkalmazást. A screen segítségével egyetlen terminálról egyszerre több munkamenetet is tudunk indítani, de egyébként is egy nagyon jó program. Minden screen által létrehozott ablak VT100-as terminálként mûködik, ezért a távoli gépen a `TERM` környezeti változó nyugodtan beállítható a `vt100` értékre.
- Tegyük hozzá a `cons25` bejegyzést a távoli gép terminálokat tároló adatbázisához. Ez pontos módszere jelentõs mértékben függ az adott gépen található operációs rendszertõl. Ebben leginkább az adott gépen található man oldalak tudnak segíteni.
- Indítsunk el a FreeBSD rendszert futtató gépen egy X szervert és a távoli géprõl egy X rendszerre íródott terminálemulátorral, például az `xterm` vagy az `rxvt` programmal jelentkezzük be. A távoli gépen ekkor a `TERM` változó értéke vagy `xterm`, vagy pedig `vt100` lesz.

5.19. A Plug and Play kártyákat miért nem találja meg (vagy unknown típusúként látja) a FreeBSD?

Ennek az okait a következõ levélben fejtette ki Peter Wemm <peter@FreeBSD.org> a FreeBSD general questions levelezési lista tagjainak, amelyben arra válaszolt, hogy egy belsõ modemet miért nem észlel a rendszer miután frissítették FreeBSD 4.*X*-re (az érthetõség kedvéért szögletes zárójelek között hozzáadtunk néhány kiegészítést is).

Az eredeti szövegbõl készült idézetet frissítettük.

A PNP BIOS beállította [a modemet] és magára hagyta valahol a portok számára fenntartott címtérben, így az ISA eszközök régi típusú [3.*X*-ben levõ] eszközpróbálgatásai ott "találták" meg.

A 4.0 esetében azonban az ISA eszközöket kezelõ kód már sokkal inkább a PnP támogatására koncentrál. Korábban [a 3.X verziókban] elõfordulhatott az is, hogy az ISA eszközök keresése során a rendszer egy "kóbor" eszközt talált, majd ugyanazt megtalálta PnP eszközként és ütköztek az így duplán lefoglalni kívánt erõforrások. Ennek kivédésére elõször tehát letiltjuk a programozható kártyák felderítését, így ez a típusú kettõs detektálás nem történhet meg. Ez továbbá azt is jelenti,

hogy a támogatott PnP hardverek azonosítóit elõre ismerni kell. Ennek hangolhatóságát már tervbevettük.

Tehát egy ilyen eszköz mûködtetéséhez szükségünk lesz a PnP azonosítójára, valamint arra, hogy felvegyük a felderítendõ PnP eszközök ISA eszközök közé. Ezt a pnpinfo(8) segítségével kérhetjük le, amely például egy belsõ modem esetén a következõ kimenetet fogja adni:

```
# pnpinfo
Checking for Plug-n-Play devices...

Card assigned CSN #1
Vendor ID PMC2430 (0x3024a341), Serial Number 0xffffffff
PnP Version 1.0, Vendor Version 0
Device Description: Pace 56 Voice Internal Plug & Play Modem

Logical Device ID: PMC2430 0x3024a341 #0
        Device supports I/O Range Check
TAG Start DF
    I/O Range 0x3f8 .. 0x3f8, alignment 0x8, len 0x8
        [16-bit addr]
    IRQ: 4  - only one type (true/edge)
```

```
TAG End DF
End Tag

Successfully got 31 resources, 1 logical fdevs
-- card select # 0x0001

CSN PMC2430 (0x3024a341), Serial Number 0xffffffff

Logical device #0
IO:  0x03e8 0x03e8 0x03e8 0x03e8 0x03e8 0x03e8 0x03e8 0x03e8
IRQ 5 0
DMA 4 0
IO range check 0x00 activate 0x01
```

Innen a `Vendor ID` kezdetû sorra lesz szükségünk. A zárójelek között szereplõ hexadecimális szám (ami a példában a `0x3024a341`) lesz az eszköz PnP azonosítója, valamint a közvetlenül ez elõtt szereplõ karakterlánc az egyedi ASCII azonosítója (`PMC2430`).

Ha a pnpinfo(8) lefuttatásának eredményeképpen megjelenõ lista nem tartalmazza a kérdéses eszközt, akkor helyette a pciconf(8) használatával is próbálkozhatunk. Íme a `pciconf -vl` parancs kimenete egy integrált hangkártya esetében:

```
# pciconf -vl
chip1@pci0:31:5:        class=0x040100 card=0x00931028 chip=0x24158086 rev=0x02 hdr
=0x00
    vendor   = 'Intel Corporation'
```

```
    device   = '82801AA 8xx Chipset AC'97 Audio Controller'
    class    = multimedia
    subclass = audio
```

Ebbõl a `chip` változót, vagyis a `0x24158086` értéket kell felhasználnunk.

Ezt az információt (a `Vendor ID` vagy a `chip` értékét) ezután a /usr/src/sys/dev/sio/sio_isa.c állományba kell felvennünk.

Ehhez elõször is készítsünk egy biztonsági másolatát a sio_isa.c állományról arra az esetre, ha véletlenül valami rossz történne. Ez azért is hasznunkra fog válni, mert így tudunk egy javítást mellékelni a hibajelentésünk mellé (mert ugye írni fogunk róla hibajelentést, ugye?). Szóval, keressük meg a sio_isa.c állományban a következõ sort:

```
static struct isa_pnp_id sio_ids[] = {
```

Menjük lentebb egészen addig, amíg nem találunk egy helyet, ahova be tudunk szúrni egy bejegyzést az eszközünkhöz. A bejegyzések megadásának módja lentebb látható, és a jobb oldalt megjegyzésbe tett ASCII Vendor ID szerint rendezettek, amelyek mellett még megtalálható (amennyiben kifér) a pnpinfo(8) *Device Description* kimenetében kapott érték is:

```
{0x0f804f3f, NULL},     /* OZO800f - Zoom 2812 (56k Modem) */
{0x39804f3f, NULL},     /* OZO8039 - Zoom 56k flex */
{0x3024a341, NULL},     /* PMC2430 - Pace 56 Voice Internal Modem */
{0x1000eb49, NULL},     /* ROK0010 - Rockwell ? */
{0x5002734a, NULL},     /* RSS0250 - 5614Jx3(G) Internal Modem */
```

A megfelelõ helyre ezután vegyük fel az eszközünkhöz tartozó hexadecimális Vendor ID értéket, mentsük el az állományt, fordítsuk újra a rendszermagot és indítsuk újra vele a rendszerünket. Ha mindent jól csináltunk, akkor az eszköz sio eszközként fog megjelenni.

5.20. Miért keletkezik nlist failed hiba például a top vagy systat parancsok futtatásakor?

A gondot alapvetõen az okozza, hogy a kérdéses alkalmazás valamiért egy olyan rendszermagbeli szimbólumot keres, amit nem talál. Ez a típusú hiba a következõkbõl eredhet:

- A rendszermag és a hozzá tartozó programok nincsenek szinkronban (vagyis fordítottunk egy új rendszermagot, de nem volt `installworld` vagy fordítva) és emiatt a szimbólumokat tároló táblázat nem teljesen úgy épül fel, ahogy azt az alkalmazás gondolja. Ha errõl lenne szó, akkor egyszerûen nincs más teendõnk, mint befejezni a frissítést (ennek pontos részleteit lásd a /usr/src/UPDATING állományban).
- Nem a `/boot/loader`, hanem közvetlenül a boot2 (lásd boot(8)) segítségével töltjük be a rendszermagot. Noha alapvetõen semmilyen problémát nem nem okoz a `/boot/loader` kihagyása, általánosságban véve azért mégis jobban elérhetõvé tudja tenni a rendszermagban

található szimbólumokat a felhasználói programok felé.

5.21. Miért tart olyan sokáig ssh vagy telnet használatával csatlakozni a számítógéphez?

A tünet: nagyon sok idõ telik aközött, amíg a TCP kapcsolat felépül és a kliens bekéri a jelszót (vagy a telnet(1) esetében amíg a bejelentkezõ képernyõ megjelenik).

A betegség: nagyon valószínû, hogy a késlekedést az okozza, amikor a szerver megpróbálja a kliens IP-címét feloldani hálózati névvé. Sok szerver, köztük a FreeBSD-ben is megtalálható Telnet és SSH szerver is ezt csinálja, többek közt azért, hogy a rendszergazda számára el tudja tárolni egy naplóban ezt a hálózati nevet.

Az orvosság: ha az említett jelenség minden olyan esetben jelentkezik, amikor a számítógéprõl (mint kliensrõl) valamilyen szerverhez csatlakozni akarunk, akkor a kliens oldalán lesz a gond. Ehhez hasonlóan, ha csak egy adott szervernél tapasztaljuk, akkor azzal a számítógéppel történhetett valami.

Amennyiben a problémákat a kliens okozza, nem tehetünk mást, a névoldáson kell úgy javítanunk, hogy a szerver normálisan fel tudja oldani. Ha helyi hálózaton tapasztaljuk mindezt, akkor ez már a szerver problémája és olvassunk tovább. Ellenkezõ esetben az internet a felelõs, ezért nagyon valószínû, hogy fel kell vennünk a kapcsolatot az internet-szolgáltatónkkal és segítséget kérni tõlük a hiba elhárításában.

Ha a problémát viszont a helyi hálózaton található szerver okozza, akkor úgy kell azt beállítanunk, hogy a helyi neveket képes legyen rendesen feloldani. Ezzel kapcsolatban a hosts(5) és named(8) man oldalakat érdemes elolvasnunk. Ha a probléma viszont az interneten jelenik meg, akkor valószínû, hogy a szerver névfeloldása nem üzemel rendesen. Nézzünk meg egy másik gépet - például a `www.yahoo.com` címet. Ha ez sem mûködik, akkor nálunk van a gond.

A FreeBSD friss telepítését követõen az is elképzelhetõ, hogy egyszerûen csak hiányoznak a tartományokkal és névszerverekkel kapcsolatos megfelelõ adatok az /etc/resolv.conf állományból. Ez gyakran okoz késlekedést az SSH mûködésében, mivel az /etc/ssh könyvtárban található sshd_config állományban alapértelmezés szerint a `UseDNS` beállítás értéke `yes` (tehát a névfeloldás használata engedélyezett). Ha valóban ez okozza a problémát, akkor a pótoljuk az /etc/resolv.conf állományból hiányzó adatokat vagy az sshd_config állományban a `UseDNS` értéke ideiglenesen legyen `no`.

5.22. Mire utal a stray IRQ (kóbor megszakítási kérés) üzenet?

A kóbor megszakítási kéréseket jelzõ üzenetek általában a hardveres megszakítási kérések egyenletlenségeire utalnak, ezen belül is leginkább olyan esetekre, amikor az eszköz egy megszakítási kérés nyugtázása közepén eltávolítja az adott kérést.

Három dolgot tehetünk ezzel kapcsolatban:

- Elviseljük ezeket a figyelmeztetéseket. Megszakítási kérésenként az elsõ öt üzenet után amúgy

sem jelez többet a rendszer.

- Ha platformunkhoz (mint például i386™) tartozó intr_machdep.c állományban található `MAX_STRAY_LOG` értékét átírjuk `5`-rõl `0`-ra és így újrafordítjuk a rendszermagot, akkor ezzel teljesen letilthatjuk a figyelmeztetéseket.
- Megszüntetjük az üzeneteket úgy, hogy csatlakoztatunk a rendszerhez egy olyan párhuzamos vonali eszközt, amely a 7-es IRQ-t használja, és rakunk fel hozzá egy PPP meghajtót (a legtöbb helyen egyébként ezzel lesz a gond), valamint a 15-ös IRQ-ra pedig rakunk egy IDE-meghajtót vagy más hasonló eszközt és telepítjük hozzá a megfelelõ meghajtót.

5.23. Miért jelenik meg folyamatosan a file: table is full üzenet a rendszernaplóban?

Ha ilyen hibaüzenetet látunk, akkor az arra utal, hogy kifogytunk a rendszerünkben egyszerre használható állományleírókból. A probléma leírásával és megoldásával kapcsolatban olvassuk el a kézikönyvben a kern.maxfiles változóról szóló részt A rendszermag korlátainak finomhangolása címû szakaszban.

5.24. Miért árasztják el calcru: negative runtime vagy calcru: runtime went backwards üzenetek a konzolt?

Ismert egy olyan probléma, hogy a BIOS-ban engedélyezzük az Intel® Enhanced SpeedStep technológiáját, akkor a rendszermag ehhez hasonló `calcru` üzeneteket kezd el küldözgetni:

```
calcru: runtime went backwards from 6 usec to 3 usec for pid 37 (pagezero)
calcru: runtime went backwards from 6 usec to 3 usec for pid 36 (vmdaemon)
calcru: runtime went backwards from 170 usec to 138 usec for pid 35 (pagedaemon)
calcru: runtime went backwards from 553 usec to 291 usec for pid 15 (swi6: task queue)
calcru: runtime went backwards from 15521 usec to 10366 usec for pid 2 (g_event)
calcru: runtime went backwards from 25 usec to 12 usec for pid 11 (swi1: net)
calcru: runtime went backwards from 4417 usec to 3960 usec for pid 1 (init)
calcru: runtime went backwards from 2084385 usec to 1793542 usec for pid 1 (init)
calcru: runtime went backwards from 408 usec to 204 usec for pid 0 (swapper)
```

Ennek oka, hogy az Intel® SpeedStep (EIST) egyes alaplapokkal nem kompatibilis.

Megoldás: Tiltsuk le a BIOS-ban az EIST használatát. Ekkor még az ACPI-alapú processzorfrekvencia-szabályozás továbbra is elérhetõ a powerd(8) használatán keresztül.

5.25. Miért jár rosszul az óra a számítógépen?

A számítógépnek kettõ vagy több idõmérõ eszköze van, és a FreeBSD pont a rosszabbikat választotta.

Adjuk ki a dmesg(8) parancsot és vizsgáljuk meg a `Timecounter` kezdetû sorokat. Ezek közül a FreeBSD a legnagyobb "quality" értékkel rendelkezõt választotta.

```
# dmesg | grep Timecounter
Timecounter "i8254" frequency 1193182 Hz quality 0
Timecounter "ACPI-fast" frequency 3579545 Hz quality 1000
Timecounter "TSC" frequency 2998570050 Hz quality 800
Timecounters tick every 1.000 msec
```

Errõl a kern.timecounter.hardware sysctl(3) változó lekérdezésével tudunk ténylegesen megbizonyosodni:

```
# sysctl kern.timecounter.hardware
kern.timecounter.hardware: ACPI-fast
```

Elõfordulhat, hogy az ACPI-idõzítõ hibás. Ilyenkor az a legegyszerûbb, ha az /etc/loader.conf állományban letiltjuk az ACPI-idõzítõ használatát:

```
debug.acpi.disabled="timer"
```

Vagy a BIOS is tudja módosítani a TSC idõzítõt - például azért, hogy csökkentse a processzor sebességét, amikor merül az akkumulátor vagy energiatakarékos módra vált. A FreeBSD sajnos nem figyel ezekre a változtatásokra és elcsúszik az idõméréssel.

Ahogy viszont az iménti példában is látható, itt még az i8254 idõzítõ is használható, méghozzá úgy, hogy a kern.timecounter.hardware sysctl(8) változó értékét átállítjuk erre az értékre:

```
# sysctl -w kern.timecounter.hardware=i8254
kern.timecounter.hardware: TSC -> i8254
```

Innentõl kezdve a számítógépünk már sokkal pontosabban mutatja az idõt.

Ezt a változtatást úgy tudjuk minden rendszerindítás során automatikusan megtenni, ha felvesszük a következõ sort az /etc/sysctl.conf állományba:

```
kern.timecounter.hardware=i8254
```

5.26. A rendszer laptopon miért nem tudja rendesen megtalálni a PC-kártyákat?

Ez a probléma gyakran megjelenik olyan laptopokon, amelyek egynél több operációs rendszert is futtatnak, egyes nem-BSD típusú rendszerek ugyanis hajlamosak a hardvert inkonzisztens állapotban hagyni. Emiatt a pccardd(8) parancs az adott kártyát "(null)""(null)" néven észleli a valós típusa helyett.

A hardvert innen teljesen csak úgy tudjuk alapállapotába hozni, ha a PC-kártya foglalatát

áramtalanítjuk. Ehhez ki kell kapcsolnunk a laptopot. (Tehát ne tegyük se készenléti, se pedig hibernált állapotba - teljesen ki kell kapcsolni.) A PC-kártya ezután várhatóan már mûködni fog.

Némely laptopok hazudnak arról, hogy rendesen ki vannak-e kapcsolva. Amennyiben az elõbbi módszer nem válna be, próbáljuk meg úgy, hogy kikapcsoljuk a gépet, kivesszük az akkumulátort, várunk egy keveset, visszarakjuk és újra bekapcsoljuk.

5.27. Miért ad a FreeBSD rendszertöltõje Read error hibát és áll meg a BIOS képernyõn?

A FreeBSD rendszertöltõje rosszul ismerte fel a merevlemez geometriáját. Ezt a FreeBSD slice-ok létrehozásakor és módosításakor külön meg kell adni az fdisk(8) használatakor.

A meghajtóhoz tartozó megfelelõ geometriai beállítások a számítógép BIOS-ában találhatóak. Keressük meg az adott meghajtó cilinder-fej-szektor (Cylinder/Head/Sector) értékét.

A sysinstall(8) partíciószerkesztõjében a `G` billentyû lenyomásával tudjuk beállítani ezt.

Ekkor egy párbeszédablak jelenik meg, ahol meg tudjuk adni a cilinderek, fejek és a sávonkénti szektorok számát. Ide perjelekkel elválasztva gépeljük e a BIOS-ban talált értékeket. Például ha a merevlemez geometriája 5000 cilinder, 250 fej és sávonként 60 szektor, akkor a `5000/250/60` értéket kell megadnunk.

Az `Enter` billentyû lenyomására ezek az értékek beállítódnak, és a `W` lenyomására pedig az új partíciós tábla kiíródik a lemezre.

5.28. Egy másik operációs rendszer letörölte a boot managert. Hogyan lehet visszaállítani?

Indítsuk el a sysinstall(8) programot, majd válasszuk a Configure és Fdisk menüpontokat. A partíciószerkesztõben a `Space` billentyûvel tudjuk kiválasztani azt a partíciót, amelyen korábban a boot manager volt. Ezután az `W` billentyû lenyomásával tudjuk a változtatásainkat lemezre menteni. Ekkor egy menü jelenik meg, ahol a telepíteni kívánt rendszertöltõt választhatjuk ki. Adjuk meg és ekkor visszakerül a helyére.

5.29. Mit jelent a swap_pager: indefinite wait buffer: hibaüzenet?

Ez arra utal, hogy egy futó program megpróbált kiírni egy lapot a memóriából a lemezre, azonban 20 másodperce már nem tudott hozzáférni a lemezhez. Ezt okozhatják hibás szektorok a lemezen, a lemez hibás kábelezése vagy esetleg valamilyen lemezmûveletekkel kapcsolatos hardver meghibásodása. Amennyiben maga a meghajtó a rossz, akkor az ilyen hibaüzenetek mellett még más, a lemez hibás mûködésére utaló üzenetet is látnunk kell a /var/log/messages állományban vagy a `dmesg` kimenetében. Minden más esetben érdemes a meghajtó csatlakozásait és kábelezését ellenõrizni.

5.30. Mik azok a UDMA ICRC hibák és hogyan lehet ellenük tenni valamit?

A ata(4) meghajtó jelenti ezeket a `UDMA ICRC` hibákat olyan esetekben, amikor a merevlemezre vagy a merevlemezrõl érkezõ DMA átvitel hibás. A meghajtó ilyenkor még párszor megpróbálja megismételni a mûveletet. Amennyiben ezek a mûveletek is meghiúsulnak, a DMA átvitel helyett a lassabb PIO átviteli módra állítja át a merevlemez felé irányuló kommunikációt.

Ezt a problémát több tényezõ is okozhatja, habár ennek a leggyakoribb oka a hibás vagy rossz kábelezés. Ilyenkor mindig ellenõrizzük, hogy a merevlemezhez csatlakozó ATA-kábelek sértetlenek és a használni kívánt Ultra DMA átviteli módra alkalmasak. Ha cserélhetõ lemezes meghajtót használunk, akkor kompatibilisnek is kell lenniük. Ez a gond akkor jelentkezhet, amikor ugyanarra az ATA-csatornára egy Ultra DMA 66-os (vagy annál is gyorsabb) és egy régebbi meghajtót csatlakoztatunk. Végezetül ezek a hibaüzenetek arra is utalhatnak, hogy a meghajtó meghibásodott. A legtöbb gyártó külön szoftver ajánl fel ennek vizsgálatára, ezért ilyenkor érdemes letesztelnünk az érintett meghajtót, illetve amennyiben szükséges, biztonsági másolatot készíteni az adatainkról és kicserélni az eszközt.

Az atacontrol(8) segédprogram használatával ellenõrizni tudjuk, hogy jelenleg az egyes ATA-eszközök milyen DMA vagy PIO módban mûködnek. Erre a célra különösen az `atacontrol mode csatorna` parancsot javasoljuk, amivel képesek vagyünk megnézni az adott ATA-csatornára csatlakozó eszközök átviteli módjait. Itt a *csatorna* értéke nullától indul.

5.31. Mi az a lock order reversal?

Erre a kérdésre a választ a FreeBSD-s szakkifejezések gyûjteményében találjuk meg a LOR címszó alatt.

5.32. Mit jelent a Called ... with the following non-sleepable locks held üzenet?

Ez az üzenet arra utalhat, hogy egy függvény lepihent miközben nála volt egy mutex (vagy más, nem pihentethetõ) típusú zárolás.

Azért keletkezik ilyen hiba, mert a mutexeket nem úgy tervezték, hogy hosszabb ideig is meg lehessen tartani, kizárólag csak rövid idõtartamra vonatkozó szinkronizációt lehet velük végezni. Ez a programozói megegyezés lehetõvé teszi az eszközmeghajtók számára, hogy a megszakítások közben mutexek segítségével képesek legyenek szinkronizálni a rendszermag többi részével. A megszakítások (FreeBSD alatt) pedig nem pihenhetnek. Ezért a rendszermagon belül semmilyen olyan alrendszer nem blokkolódhat huzamosabb ideig, amelyik mutexet tart magánál.

Ezeket a hibákat úgy tudjuk elcsípni, ha olyan ellenõrzéseket teszünk a rendszermagba, amelyek jeleznek a witness(4) alrendszernek, hogy küldjön figyelmeztetést vagy akár végzetes hibát (a rendszer konfigurációjától függõen) azokban a helyzetekben, amikor egy sejthetõen hosszabb ideig blokkolt hívás tart magánál egy mutexet.

Röviden úgy foglalhatnánk össze, hogy ezek a hibák alapvetõen nem végzetesek, de egy kis

balszerencsével az egyszerû kis megakadásoktól kezdve a teljes lefagyásig szinte bármilyen hibáért felelõsek lehetnek.

5.33. A buildworld/installworld miért áll le touch: not found hibával?

Ez a hibaüzenet nem azt jelenti, hogy a touch(1) segédprogram nem található, hanem inkább azt, hogy az érintett állományok dátuma a jövõre állítódott be. Ha a CMOS óránkat a helyi idõ szerint állítottuk be, akkor egyfelhasználós módban indítsuk újra a rendszert és a `adjkerntz -i` parancs kiadásával állítsuk be a rendszermag óráját.

Chapter 6. Kereskedelmi alkalmazások

Ez a fejezet még nagyon rövid, de természetesen reméljük, hogy a különbözõ cégek hamarosan bõvíteni fogják! :) A FreeBSD fejlesztõinek ezzel kapcsolatban semmilyen anyagi érdekük nincs, csupán szeretnék felsorolni a nyilvánosan is elérhetõ szolgáltatásokat (de úgy érezzük, hogy a FreeBSD kereskedelmi irányú megközelítése a FreeBSD fejlõdésére is jó hatással lehet hosszabb távon). Javasoljuk minden kereskedelmi fejlesztõnek, hogy küldjék be ide is a saját kérdéseiket. A gyártók honlapján olvashatjuk a teljes listájukat.

6.1. Honnan lehet a FreeBSD-hez irodai programcsomagokat szerezni?

A nyílt forráskódú OpenOffice.org irodai programcsomag FreeBSD alatt natívan is futtatható. Oracle Open Office linuxos változata, amely az OpenOffice.org zárt forráskódú, továbbfejlesztett változata, szintén mûködik FreeBSD alatt.

A FreeBSD ezeken kívül még számos szövegszerkesztõt, táblázatkezelõt és rajzprogramot is tartalmaz a Portgyûjteményében.

6.2. Honnan lehet a Motif®-ot szerezni a FreeBSD-hez?

A The Open Group kiadta a Motif® 2.2.2 változatának forráskódját. Ez az x11-toolkits/open-motif csomagból vagy portból érhetõ el. A telepítésével kapcsolatban olvassuk el a kézikönyv portokról szóló részét.

Az Open Motif® kizárólag csak nyílt forráskódú operációs rendszereken terjeszthetõ.

Ezenkívül még használhatóak a Motif® kereskedelmi változatai is. Ezek viszont már nem ingyenesek, de a licencük megengedi azt, hogy zárt forráskódú szoftverekben is felhasználhassuk. Az Apps2gonál érdeklõdjünk a FreeBSD-re elérhetõ legolcsóbb Motif® 2.1.20 ELF (i386™) típusú terjesztésekkel kapcsolatban.

Kétfajta terjesztés létezik, a "fejlesztõi változat" és a "futásidejû változat" (valamivel olcsóbb). Az egyes terjesztésekben a következõk találhatóak:

- OSF/Motif® manager, xmbind, panner, wsm
- Fejlesztõi készlet: uil, mrm, xm, xmcxx, include és Imake állományok
- Statikus és dinamikus ELF könyvtárak
- Példa alkalmazások

A megrendelés során ne felejtsük el megadni, hogy a Motif® melyik FreeBSD verzióhoz készített változatát kérjük (valamint az architektúrát se)! Az *Apps2go* NetBSD és OpenBSD rendszerekkel is foglalkozik, ezeket a változatokat jelenleg csak FTP-n keresztül lehet elérni.

További információk

Az Apps2go honlapja

illetve

sales@apps2go.com vagy support@apps2go.com

vagy

telefonon: (817) 431 8775 és +1 817 431-8775

6.3. Honnan lehet FreeBSD-re CDE-t szerezni?

A *Xi Graphics* korábban kínált fel CDE-t FreeBSD-hez, de manapság már nem foglalkoznak ezzel.

A KDE a CDE-hez nagyon sok tekintetben hasonló nyílt forráskódú X11 munkakörnyezet, de érdemes pillanatást vetnünk az xfce-re is. A KDE és az xfce egyaránt megtalalálható a portok között.

6.4. Használhatóak adatbázisrendszerek FreeBSD alatt?

Igen! A FreeBSD hivatalos honlapján megtaláljuk ezeket a kereskedelmi gyártók között.

Érdemes még megnéznünk a Portgyûjteményeben a adatbázisokat tartalmazó szekciót.

6.5. Az Oracle® fut FreeBSD alatt?

Igen. A http://www.shadowcom.net/freebsd-oracle9i/ oldalon találunk arról információt, hogyan telepíthetjük FreeBSD-re az Oracle® Linux® változatát.

Chapter 7. Felhasználói alkalmazások

7.1. Hol vannak a felhasználói programok?

Nézzünk szét a portok között és láthatjuk, hogy milyen szoftvereket portoltak eddig FreeBSD-re. A listában pillanatnyilag 36000 port található és naponta növekszik, ezért érdemes folyamatosan figyelni vagy az új portokról úgy is értesülhetünk rendszeresen, ha feliratkozunk a FreeBSD announcements levelezési lista címére.

A legtöbb portnak mûködnie kell a 6.*X*, 7.*X* és 8.*X* ágak használata esetén is. Mindegyik FreeBSD kiadás elkészítésekor készül egy pillanatfelvétel a portokat tartalmazó könyvtárról és bekerül a ports/ könyvtárba.

Ezenkívül még "csomagok" is rendelkezésünkre állnak, amelyek lényegében egy tömörített bináris terjesztési formát takarnak, némi plusz információval kiegészítve az egyéni telepítésekhez elvégzéséhez. A csomagok könnyen telepíthetõek és eltávolíthatóak anélkül, hogy pontosan ismernénk a benne található állományok összes apró részletét.

A különbözõ csomagokat a sysinstall(8) programban (a Configure menün belül) található Packages menüpontban tudjuk telepíteni, vagy meghívjuk meg a pkg_add(1) parancsot. A csomagokat leginkább .tbz kiterjesztésükrõl lehet megismerni, valamint a telepítõ CD-ken a packages/All könyvtárban találhatóak. Az interneten keresztül is le tudjuk tölteni ezek közül a FreeBSD különbözõ verzióihoz tartozó változatukat a hozzánk legközelebbi tükrözésekrõl:

6.*X*-RELEASE/6-STABLE esetén

ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/i386/packages-6-stable/

7.*X*-RELEASE/7-STABLE esetén

ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/i386/packages-7-stable

8.*X*-RELEASE/8-STABLE esetén

ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/i386/packages-8-stable

9-CURRENT esetén

ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/i386/packages-9-current

Nem mindegyik port érhetõ el csomagként, mivel folyamatosan készülnek az újabbak. Ezért mindig érdemes bizonyos idõközönként ellenõrizni a központi ftp.FreeBSD.org oldalon található csomagokat.

7.2. Hogyan tudjuk beállítani az INN (Internet News) szolgáltatást a gépünkön?

Telepítsük az news/inn csomagot vagy portot és utána kiindulásképpen nézzük meg az INN GYIK-ot.

7.3. A FreeBSD rendelkezik Java™ támogatással?

Igen. Látogassunk el a http://www.FreeBSD.org/java/ oldalra.

7.4. Miért nem fordul egy port a 6.X-STABLE, 7.X-STABLE vagy 8.X-STABLE változatot futtató gépeken?

Ha olyan FreeBSD verziónk van, amely egy kicsit lemaradt az aktuális *-CURRENT* vagy *-STABLE* ágak mögött, akkor valószínûleg frissítenünk kell a Portgyûjteményünket. Ennek részleteirõl a Porterek kézikönyvében, a Keeping Up címû részben olvashatunk (angolul). Ha viszont rendszerünkben minden a lehetõ legfrissebb, akkor elõfordulhat, hogy valaki olyan változtatást rakott fel a porthoz, amely a *-CURRENT* esetén mûködik, de a *-STABLE* változatban már nem. Ilyenkor feltétlenül küldjünk egy hibajelentést a send-pr(1) paranccsal, hiszen a Portgyûjteménynek a -CURRENT és -STABLE ágak esetén egyaránt mûködnie kell.

7.5. A make index paranccsal nem sikerült létrehozni az INDEX állomyánt. Mi a gond?

Elsõként mindig ellenõrizzük, hogy a Portgyûjteményünk a lehetõ legfrissebb. A legfrissebb változatnál jelentkezõ INDEX készítési hibák mindig szem elõtt vannak, ezért általában gyorsan megjavulnak.

Ha viszont egy friss verzióval rendelkezünk, akkor elképzelhetõ, hogy egy másik hibával kerültünk szembe. A `make index` parancsnak van egy olyan hibája, amely miatt nem képes a Portgyûjtemény hiányos példányával dolgozni. Feltételezi ugyanis, hogy az összes olyan port megtalálható a rendszerünkben, amely telepítése szükséges az adott porthoz. Ennek megértéséhez most képzeljük el, hogy megvan az ize/mize port a lemezen, amely függ az aze/maze porttól, és emiatt az aze/maze portnak és függõségeinek is rajta kell lennie a lemezünkön. Minden más esetben a `make index` nem tud összegyûjteni elegendõ információt ahhoz, hogy létre tudja hozni a függõségi gráfot.

Ez különösen olyan FreeBSD felhasználókkal fordul elõ, akik a cvsup(1) (vagy csup(1)) használatával frissítik a Portgyûjteményüket, de a refuse állományokban kizártak néhány kategóriát. Elméletben természetesen ki lehet zárni akármilyen kategóriát, azonban a gyakorlat azt mutatja, hogy ez szinte lehetetlen, mivel túlságosan sok port függ más kategóriákban található portoktól. Amíg valaki meg nem oldja ezt a problémát, addig fogadjuk el általános szabálynak, hogy az INDEX létrehozásához a teljes Portgyûjteménnyel rendelkeznünk kell.

Néhány ritka esetben még elõfordulhat, hogy az INDEX azért nem jön létre, mert a make.conf állományban megadtunk valamilyen `WITH*` vagy `WITHOUT*` változót. Ha úgy érezzük, hogy ez okozhatja a problémát, akkor próbáljuk meg elõször ezen változók nélkül létrehozatni az INDEX állományt és csak utána jelenteni a hibát a FreeBSD ports levelezési lista címére.

7.6. A CVSup miért nincs a FreeBSD forrásai között?

A FreeBSD alaprendszerét úgy állították össze, hogy saját magát legyen képes legyen lefordítani, vagyis az egész operációs rendszer elõállítható legyen néhány alapvetõ eszköz használatával. Ezért

a források között leginkább csak az található meg, ami feltétlenül kell a források lefordításához. Ilyen például a C fordító (gcc(1)), a make(1), awk(1) és a többi.

Mivel a CVSup a Modula-3 programozási nyelven íródott, csak úgy tudnánk beletenni a FreeBSD alaprendszerbe, ha hozzávennénk és karbantartanánk egy Modula-3 fordítót is. Ezzel együtt viszont növekedne a FreeBSD forrása, amelyet aztán karban is kellene tartani. Ezért mind a fejlesztõk, mind pedig a felhasználók számára egyszerûbb, ha a CVSup egy külön portként érhetõ el a rendszerhez. Ez viszont gyorsan telepíthetõ a FreeBSD telepítõ CD-ken található csomagokból.

Azonban a FreeBSD 6.2-RELEASE megjelenésétõl kezdve a FreeBSD felhasználók nem maradnak integrált CVSup kliens nélkül. Maxime Henrion <mux@FreeBSD.org> munkájának köszönhetõen a CVSup alkalmazásnak elkészült a C nyelven újraírt változata, a csup(1), amely most már az alaprendszer része. Noha jelenleg nem még nem képes mindarra, amire a CVSup, elegendõ (és nagyon gyors!) ahhoz, hogy a forrásainkat frissen tartsuk. A 6.2 elõtt kiadott rendszerek esetében ezt portból vagy csomagból is felrakhatjuk (lásd net/csup).

7.7. A forrásokon kívül a telepített portokat is lehet valahogy frissíteni?

A FreeBSD alaprendszere ehhez nem kínál fel semmilyen eszközt, de léteznek olyan segédeszközök, amelyekkel valamennyire meg tudjuk könnyíteni a frissítés folyamatát. További segédprogramok telepítésével pedig a portok kezelését tudjuk tovább egyszerûsíteni, amirõl a FreeBSD kézikönyv A portok frissítése címû szakaszában olvashatunk bõvebben.

7.8. Minden nagyobb verziófrissítésnél újra kell fordítani az összes telepített portot a rendszeren?

Mindenképpen! Noha látszólag a frissített rendszeren is remekül futnak a korábbi verzióra telepített alkalmazások, könnyen elõfordulhat, hogy az újabb portok telepítésékor vagy a meglevõek frissítésekor véletlenszerû összeomlásokat vagy egyéb hibákat tapasztalunk.

Ne felejtsük el, hogy a rendszer frissítésekor a különféle osztott könyvtárak, betölthetõ modulok és a rendszer egyéb komponensei is lecserélõdnek. Ezért a régebbi változataikhoz fordított alkalmazások egyáltalán nem fognak elindulni vagy nem mûködnek rendesen.

Ezzel kapcsolatban olvassuk el a FreeBSD kézikönyvének frissítérõl szóló szakaszát.

7.9. Minden kisebb verziófrissítésnél újra kell fordítani az összes telepített portot a rendszeren?

Általánosságban véve nem. A FreeBSD fejlesztõi ugyanis mindent megtesznek azért, hogy ugyanazon a fõ fejlesztési ágon belüli verziók között megmaradjon a bináris szintû kompatibilitás. Az esetleges kivételeket pedig dokumentálni szokták a kiadásokhoz tartozó jegyzetekben, ahol többnyire megadják az adott változtatáshoz tartozóan a követendõ tanácsokat.

7.10. A /bin/sh miért ilyen egyszerû? A FreeBSD-ben miért nincs bash vagy valamilyen más rendes parancsértelmezõ?

Mert a POSIX® szerint lennie kell egy ilyen parancsértelmezõnek.

A valamivel bonyolultabb válasz: sokan szeretnének olyan szkripteket írni, amelyek több rendszer közt is átvihetõek. Ezért a POSIX® a parancsértelmezõkre és a segédprogramokra vonatkozó parancsokat igen részletesen tárgyalja. A legtöbb ilyen szkriptet a Bourne-féle parancsértelmezõben készítik, és több fontos programozói felület (make(1), system(3), popen(3) és ezek magasabb szintû, például Perl és Tcl nyelvi megfelelõi) a Bourne-parancsértelmezõ használatán alapszik. Mivel a Bourne-parancsértelmezõ használata ilyen széles körben elterjedt, fontos, hogy gyorsan induljon, elõre megjósolható legyen a mûködése és ne foglaljon túlságosan sok memóriát.

A jelenlegi implementáció igyekszik ezek közül az elvárások közül egyszerre a lehetõ legtöbbet teljesíteni. A `/bin/sh` programot csak úgy tudjuk a megfelelõ méreten tartani, ha nem tesszük bele az összes többi parancsértelmezõben megtalálható kényelmi funkciót. Pontosan ezért találhatjuk meg viszont a Portgyûjteményben a többi, például a `bash`, `scsh`, `tcsh` és `zsh` parancsértelmezõket. (Ezek konkrét memóriahasználatát össze is tudjuk vetni, ha a `ps -u` parancs kimenetének "VSZ" és "RSS" oszlopait megnézzük.)

7.11. A getenv(3) és az Opera indítása miért tart olyan sokáig?

Erre az az általános válasz, hogy a névfeloldás valószínûleg rosszul mûködik a rendszerünkön. A getenv(3) és az Opera is ellenõrzi a névfeloldást az indulásakor. Ezért a böngészõ egészen addig nem jelenik meg az asztalon, amíg választ nem kap vagy rá nem jön, hogy nincs aktív hálózati kapcsolat.

7.12. Ha a CVSup használatával frissítjük a Portgyûjteményt, akkor sok port nem fordul le mindenféle rejtélyes hibaüzenet kíséretében! Valami nagy baj van a Portgyûjteménnyel?

Ha úgy korábban úgy frissítettük a CVSup használatával a Portgyûjteményt, hogy nem adtuk meg a `ports-all` CVSup algyûjteményt, akkor a `ports-base` algyûjteményt is *mindig* frissítenünk kell! Ennek okairól a kézikönyvben olvashatunk.

7.13. Hogyan lehet MIDI állományokból audio CD-t készíteni?

Ha MIDI állományokból akarunk audio CD-t készíteni, akkor elõször telepítsük fel a

Portgyûjteménybõl a audio/timidity portot, majd kézzel tegyük hozzá Eric A. Welsh GUS patch-eit, melyek a http://alleg.sourceforge.net/digmid.html címrõl tölthetõek le. Miután a TiMidity++ sikeresen felkerült a rendszerünkre, a MIDI állományokat a következõ paranccsal tudjuk átkonvertálni WAV állományokra:

```
% timidity -Ow -s 44100 -o /tmp/juke/01.wav 01.mid
```

A WAV állományok ezek után tetszõleges formátumba konvertálhatóak tovább vagy készíthetõ belõlük egy audio CD, ahogy azt a FreeBSD kézikönyvben is olvashatjuk.

Chapter 8. A rendszermag beállítása

8.1. Nehéz testreszabni a rendszermagot?

Egyáltalán nem! Ezzel kapcsolatban olvassuk el a FreeBSD kézikönyv rendszermag beállításairól szóló részét.

Az új kernel állomány a hozzá tartozó modulokkal együtt a /boot/kernel könyvtárba települ, míg a rendszermag korábbi változata és a moduljai a /boot/kernel.old könyvtárba kerül át, így ha netalán valamit elrontottunk volna, akkor a rendszermag korábbi változatának betöltésével lehetõségünk lesz kijavítani a hibát.

8.2. A rendszermag nem fordul le, mert a _hw_float nem található. Hogyan lehet megoldani ezt a problémát?

Ez valószínûleg azért következett be, mert eltávolítottuk az npx0 (lásd npx(4)) támogatást a rendszermag beállításai közül, mert a rendszerünkben nincs matematikai társprocesszor. Az npx0 eszköz jelenléte azonban *kötelezõ*. Valahol a gépünkben lennie kell olyan eszköznek, amely a lebegõpontos számok hardveres kezelését végzi, annak ellenére, hogy nem egy különálló eszköz, ahogy régen a 386-osoknál volt. A rendszermagban szerepelnie *kell* az npx0 eszköznek. Ha netalán még sikerülne is npx0 támogatás nélkül fordítanunk egy rendszermagot, akkor sem tud elindulni.

8.3. Miért ilyen nagy a rendszermag mérete (közel 10 MB)?

Nagyon valószínû, hogy a rendszermagunk *debug módban* készült el. A debug módú rendszermagokban rengeteg olyan szimbólum található, amely hasznos lehet a hibák keresése és a rendszer vizsgálata során, ezért emiatt jelentõs mértékben növekszik a mérete. Emiatt nem kell aggódnunk, mert egy hibakeresésre felkészített rendszermag egyáltalán nem vagy csak egy kicsivel lassabb, mint a hagyományos változat, illetve a rendszer összeomlásakor mindig mindig szükségünk lehet ezekre a debug információkra.

Ha viszont kevés a lemezterület vagy egyszerûen csak nem akarunk debug módú rendszermagot akarunk futtatni, akkor a következõkre kell figyelnünk:

- Vegyük ki a rendszermag konfigurációs állományából a következõ sort:

```
makeoptions DEBUG=-g
```

- A config(8) használata során ne használjuk a `-g` beállítást.

A fentiek közül akármelyiket is választjuk, a rendszermagunk debug módban jön létre. Ha azonban

sikerült betartani a fentebb javasolt lépéseket, akkor egy normál rendszermagot kapunk, amely mérete ilyenkor jelentõs mértékben visszaesik: a legtöbbjük olyan 1,5 és 2 MB körül van.

8.4. Miért ütköznek a megszakítások, amikor többportos soros vonali kártyákat akarunk használni?

Ha a rendszermagot a többportos soros vonali kártyák támogatásával fordítjuk le, akkor a rendszertõl azt az üzenetet kapjuk, hogy csak az elsõ megszakítást fogja használni, a többit pedig ütközés miatt (interrupt conflict) kihagyja. Hogyan lehet ezen javítani?

A gondot alapvetõen az okozza, hogy a FreeBSD a rendszermagban fixen letárolja ezeket, nehogy valamilyen hardveres vagy szoftveres ütközés miatt elkallódjanak. Ezen úgy tudunk segíteni, ha egyetlen IRQ vonal kivételével az összes többi beállítását szabadon hagyjuk. Íme erre egy példa:

```
#
# Többportos nagysebességû soros vonali eszközök - 16550 UART
#
device sio2 at isa? port 0x2a0 tty irq 5 flags 0x501 vector siointr
device sio3 at isa? port 0x2a8 tty flags 0x501 vector siointr
device sio4 at isa? port 0x2b0 tty flags 0x501 vector siointr
device sio5 at isa? port 0x2b8 tty flags 0x501 vector siointr
```

8.5. Miért nem lehet lefordítani a rendszermagot, még a GENERIC beállításaival sem?

Ennek több oka is lehet. Ezek közül néhány, de nem feltétlenül ebben a sorrendben:

- Nem a `make buildkernel` és `make installkernel` parancsokat használtuk és valószínûleg a forrásaink sem egyeznek meg a jelenleg futó rendszerével (például egy 12.0-RELEASE rendszert akarunk fordítani egy 11.2-RELEASE rendszeren). Ha frissíteni akarunk, akkor olvassuk el a /usr/src/UPDATING állományt, különös tekintettel a végén található "COMMON ITEMS" címû szakaszra.
- A `make buildkernel` és `make installkernel` parancsokat használtuk, de elõtte nem futott le rendesen a `make buildworld` parancs. A `make buildkernel` parancs ugyanis erõsen támaszkodik a `make buildworld` által végzett munkára.
- Gyakran a FreeBSD-STABLE változat használata esetén is elõfordulhat, hogy olyan pillanatban töltöttük le a forrásokat, amikor módosítás alatt voltak vagy valamiért nem mûködtek rendesen. Kizárólag a kiadások esetén tudjuk szavatolni a hibátlan fordítást, noha a FreeBSD-STABLE verzióból készült változatok is többnyire megfelelõek. Próbáljuk meg újra letölteni a forrásokat, ha eddig még nem próbálkoztunk volna vele, és nézzük meg, hogy ez segít-e megoldani a problémát. Keressük másik szervert, ha gondjaink vannak a frissítéssel.

8.6. Honnan tudhatjuk meg milyen ütemezõvel dolgozik a rendszerünk?

Nézzük meg, hogy a rendszerünkben elérhetõ-e a `kern.sched.quantum` változó. Ha van ilyenünk, akkor valami ilyesmit kell tapasztalnunk:

```
% sysctl kern.sched.quantum
kern.sched.quantum: 99960
```

Ha létezik a `kern.sched.quantum` nevû sysctl változó, akkor a 4BSD ütemezõ fut (lásd sched_4bsd(4)). Ha nem, akkor egy ilyen hibát kapunk a sysctl(8) parancstól (ezt nyugodtan figyelmen kívül hagyhatjuk):

```
% sysctl kern.sched.quantum
sysctl: unknown oid 'kern.sched.quantum'
```

Az aktuálisan használt ütemezõ neve közvetlenül elérhetõ a `kern.sched.name` sysctl változó lekérdezésén keresztül:

```
% sysctl kern.sched.name
kern.sched.name: 4BSD
```

8.7. Mi az a kern.sched.quantum?

A `kern.sched.quantum` értéke határozza meg, hogy egy futó program legfeljebb mennyi órajelet futhat egyszerre, megszakítás nélkül. Ezt az értéket a 4BSD ütemezõ használja, ezért a jelenlétébõl vagy hiányából következtetni tudunk a pillanatnyilag használatban levõ ütemezõre.

Chapter 9. Lemezek, állományrendszerek és rendszertöltők

9.1. Hogyan adjunk lemezeket a FreeBSD rendszerünkhöz?

Ezzel kapcsolatban olvassuk el a lemezek hozzáadásáról szóló részt a FreeBSD kézikönyvben.

9.2. Hogyan lehet átteni a rendszert egy nagyobb lemezre?

Ezt legegyszerûbben úgy tudjuk megcsinálni, ha újratelepítjük az operációs rendszert az új lemezre és külön áttesszük a felhasználói adatokat. Ez különösen ajánlott abban az esetben, ha már több kiadás óta követjük a *-STABLE* változatot, vagy ha korábban már frissítettük a kiadásunkat. A boot0cfg(8) segítségével fel tudjuk rakni a booteasyt mind a két lemezre és így egészen addig váltogatni tudjuk a kettõt, amíg teljesen át nem álltunk. Ugorjuk át a következõ bekezdést, és olvassuk el, hogy rakjuk át az adatokat.

Úgy is dönthetünk, hogy nem telepítjük újra a rendszert. Ekkor vagy a sysinstall(8), vagy pedig a fdisk(8) és a disklabel(8) használatával osszuk fel és címkézzük meg az új lemezt. Érdemes még a boot0cfg(8) segítségével felraknunk a booteasyt mind a két lemezre, így miután átmásoltuk a régi rendszerünket az új lemezre, ennek megtartásával ki tudjuk próbálni az új rendszert.

Most, miután sikeresen beállítottuk az új lemezt, készen állunk az adatok átmásolására. Sajnos nem lehet csak úgy vakon átmásolni ezeket egyik lemezrõl a másikra. Ilyenkor ugyanis bizonyos dolgok (például a /dev könyvtárban található eszközleírók, az állományjelzõk és a linkek stb.) hajlamosak elromlani. Ezért ehhez olyan eszközökre lesz szükségünk, amelyek ismerik ezeket a dolgokat, mint például a dump(8). Továbbá javasoljuk, hogy egyfelhasználós módban végezzük el az átvitelt, noha ez nem feltétlenül szükséges.

A rendszerindító állományrendszer átmozgatásához egyedül a dump(8) és restore(8) segédprogramokra lesz szükségünk. Esetleg a tar(1) parancs is használható, de nem minden esetben. A dump(8) és restore(8) páros akkor is remekül használható, ha egy partíció tartalmát egy üres partícióra akarjuk átvinni. A következõ lépések szükségesek ahhoz, hogy a `dump` parancs segítségével átvigyük egyik partícióról a másikra az adatokat:

1. Hozzunk létre egy új partíciót.
2. Ideiglenesen csatlakoztassuk egy könyvtárba.
3. Lépjünk be abba a könyvtárba.
4. Mentsük le a régi partíciót és az eredményt küldjük át az újra.

Például, ha a /mnt könyvtárba csatlakoztatott /dev/ad1s1a eszközrõl akarjuk átvinni a jelenlegi gyökérpartíciónkat, akkor ezeket a parancsokat kell kiadnunk:

```
# newfs /dev/ad1s1a
# mount /dev/ad1s1a /mnt
# cd /mnt
# dump 0af - / | restore rf -
```

További munkát igényel, ha a dump parancs segítségével a partícióinkat is át akarjuk szervezni. Például a /var partíciót úgy tudjuk beleolvasztani a tövébe, ha létrehozunk egy olyan partíciót, amely mind a kettõ számára elegendõ nagy, majd a fentebb leírt módszerrel elõször átmozgatjuk a tövét, utána pedig átmozgatjuk az alpartíció tartalmát az elsõ mozgatás során létrejött egyik üres könyvtárba:

```
# newfs /dev/ad1s1a
# mount /dev/ad1s1a /mnt
# cd /mnt
# dump 0af - / | restore rf -
# cd var
# dump 0af - /var | restore rf -
```

Egy könyvtárat, például /var tartalmát pedig úgy tudunk leválasztani a tövérõl, vagyis átrakni egy korábban nem létezõ partícióra, ha elõször létrehozzuk mind a két partíciót, csatlakoztatjuk a leendõ alpartíciót az ideiglenes csatlakozási ponton belül a megfelelõ könyvtárba és mindkettõre átmozgatjuk a régi partíció teljes tartalmát:

```
# newfs /dev/ad1s1a
# newfs /dev/ad1s1d
# mount /dev/ad1s1a /mnt
# mkdir /mnt/var
# mount /dev/ad1s1d /mnt/var
# cd /mnt
# dump 0af - / | restore rf -
```

A felhasználói adatok esetén a cpio(1), pax(1), tar(1) és dump(8) eszközöket ajánljuk. Az írás pillanatában még úgy tudjuk, hogy nem tartják meg az állományjelzõkkel kapcsolatos információkat, ezért csak óvatosan használjuk ezeket!

9.3. A Veszélyesen dedikált (Dangerously Dedicated) lemezek veszélyesek a felhasználóra?

A telepítés során két különbözõ módon tudjuk partícionálni a lemezeinket. Alapértelmezés szerint a rendszer igyekszik kompatbilis maradni a gépünkön található többi operációs rendszerrel. Ilyenkor normális partíciós táblabeli bejegyzéseket készít (amelyeket FreeBSD alatt "slice"-oknak hívnak), és egy ilyen slice-ba teszi az összes saját partícióját. Emellé még telepíteni tudjuk az operációs rendszerek választásának lehetõségét is a rendszer indításakor. A másik lehetõség választása esetén azonban a FreeBSD teljesen kisajátítja a lemezt és nem is próbál meg kompatibilis maradni a többi operációs rendszerrel.

Miért is nevezzük ezt "veszélyesnek"? A lemez ebben az esetben nem tartalmaz semmi olyat, amelyet a hétköznapi programok partíciós táblaként tudnának beazonosítani. Attól függõen, hogy mennyire illedelmes, egy ilyen program panaszkodni fog, amikor megpróbálja értelmezni a lemez tartalmát, de rosszabb esetben anélkül felülírja a rendszerbetöltõt, hogy bármit is jelzett volna. Ráadásul a "veszélyesen dedikált módon" kiosztott lemezek még bizonyos BIOS-okat is képesek megzavarni, többek közt az Award (például amelyek a HP NetServer, Micronics és hasonló rendszerekben találhatóak) vagy Symbios/NCR (népszerû 53C8xx SCSI-vezérlõk) típusúak esetén találkozhatunk ezzel a problémával. Ez a lista persze nem teljes, más gyártók termékeivel is gondok akadhatnak. Ennek a hibának jellemzõ tünete a `read error` hibaüzenet, amely arra utal, hogy a FreeBSD betöltõje nem találja saját magát a lemezen, vagy éppen az egész rendszer megáll a rendszer indítása közben.

Akkor mégis mi értelme van ennek? Csak néhány kilobyte-tot spórolunk vele, miközben komoly gondokat okozhat egy frissen telepített rendszer esetében. A "veszélyesen dedikált mód" eredetileg az új FreeBSD telepítõket veszélyeztetõ egyik komoly hibát szeretné kiküszöbölni: a merevlemezek BIOS és lemez önmaga által ismert geometriai beállításainak egyeztetése.

A "lemezgeometria" fogalma tulajdonképpen már egy elavult fogalom, de a PC-k BIOS-a legbelül még mind a mai napig így kommunikál a lemezekkel. Amikor a FreeBSD telepítõjével slice-okat hozunk létre, olyan módon kell rögzítenünk a lemezre ezek pozícióját, hogy a BIOS képes legyen megtalálni. Ha ez nem sikerül, akkor nem tudjuk elindítani a rendszert.

A "veszélyesen dedikált" mód ezt a problémát az egyszerûsítésén keresztül próbálja megoldani, és néha sikerül is neki. Ezt azonban csak akkor javasoljuk, ha semmi más nem mûködik, hiszen az esetek túlnyomó részében más megoldás is létezik.

Hogy tudjuk tehát akkor elkerülni a "veszélyesen dedikált" mód használatát a telepítés során? Jegyezzük fel, hogy mik a BIOS szerint a merevlemezünk geometriai beállításai. Ezt a rendszerindítás közben a rendszermagtól is megkérdezhetjük úgy, hogy a `boot:` paranccsorába megadjuk a `-v` beállítást, vagy a betöltõben a `boot -v` parancsot használjuk. Így pontosan a telepítõ indítása elõtt a rendszermag ki fogja írni a BIOS által ismert geometriai beállításokat. Ne essünk pánikba, várjuk meg, amíg a telepítõ elindul, tekerjünk vissza a számokhoz és olvassuk le ezeket. A lemezek általában a BIOS sorrendjében jelennek meg, tehát elõször az IDE aztán a SCSI típusúak.

A lemez partícionálásakor ellenõrizzük, hogy az FDISK képernyõjén megjelenõ geometriai beállítások megfelelõek (tehát egyeznek a BIOS által ismert értékekkel). Ha eltérést tapasztalunk, akkor a `G` billentyû lenyomásával tudjuk átjavítani. Erre leginkább akkor lesz szükségünk, ha a lemez teljesen üres, vagy ha a lemezt egy másik rendszerbõl hoztuk át. Ez egyébként csak azoknál a lemezeknél okoz gondot, amelyekrõl a rendszert akarjuk indítani, a FreeBSD a többi lemezzel már remekül elboldogul.

Miután sikerült egyeztetnünk a BIOS és a FreeBSD geometriai beállításait, szinte biztos, hogy nem kell már emiatt aggódnunk, így a "veszélyesen dedikált" módra sincs szükségünk. Ha viszont mégis egy `read error` hibaüzenetet kapnánk a rendszer indítása közben, akkor tegyünk egy próbát. Semmit sem veszíthetünk.

Ha a "veszélyesen dedikált" mód használatáról szeretnénk visszatérni a megszokottra, akkor két lehetõségünk van. Elõször is teljesen le kell nulláznunk az MBR-t, így biztosra vehetjük, hogy az ezután következõ telepítések során egy teljesen üres lemezt látunk. Ezt például így lehet megtenni:

```
# dd if=/dev/zero of=/dev/rda0 count=15
```

A másik módszer egy hivatalosan nem dokumentált DOS-os "lehetõség" használata:

```
C:\> fdisk /mbr
```

Ezzel egy új Master Boot Recordot tudunk telepíteni, ami ezzel együtt felülírja a BSD rendszertöltõjét is.

9.4. Milyen partíciókon lehet Soft Updatest használni? A Soft Updates állítólag nem mûködik rendesen a gyökérpartíció esetén.

A rövid válasz: A Soft Updates bármelyik partíción minden további nélkül használható.

A hosszabb válasz: Korábban voltak bizonyos kétségek afelõl, hogy a Soft Updates jól mûködik a rendszerindító partíciókon is. Ez alapvetõen a Soft Updates két jellemzõjére vezethetõ vissza. Elõször is, a Soft Updatest alkalmazó partíciók esetén elõfordulhat, hogy a rendszerösszeomlás során elveszik valamennyi adat (maga a partíció nem lesz hibás, csupán némi adat tûnik el), illetve a Soft Updates ideiglenesen kifogyhat a tárhelybõl.

A Soft Updates használata során a rendszermag legfeljebb harminc másodperc múlva írja ki fizikailag a változtatásokat a lemezre. Tehát amikor egy nagyobb állományt törlünk, akkor ez az állomány egészen addig a lemezen marad, amíg a rendszermag ténylegesen el nem végzi ezt a törlést. Ezzel viszont nagyon egyszerûen létrehozható egy "ütközés" (race condition) az állományrendszeren. Tegyük fel, hogy letörlünk egy nagyobb állományt és utána közvetlenül létrehozunk egy másik nagyobb állományt. Mivel az elsõ állomány ilyenkor még nem törlõdik le valójában a lemezrõl, ezért a második számára már nem lesz elegendõ helyünk. A rendszer ekkor egy hibaüzenetben fog figyelmeztetni minket, miközben pontosan az imént töröltünk le egy óriási állományt! Ha néhány másodperccel késõbb újra megpróbáljuk létrehozni ezt az állományt, akkor már minden a megfelelõ módon fog zajlani. Ezt régebben sok FreeBSD felhasználó nem tudta mire vélni.

Ha a rendszerünk olyankor omlik össze, amikor a rendszermag már elkezdte egy nagyobb mennyiségû adat kiírását a lemezre, de még nem fejezõdött be, akkor könnyen elõfordulhat, hogy ez az adat elveszik vagy meghibásodik. Ennek kockázata nagyon kicsi, és általában kezelhetõ, viszont az IDE-meghajtókban található írási gyorsítótár használata jelentõsen növeli ezt. Ezért a Soft Updates alkalmazása során nem javasoljuk ennek használatát.

Ezek a problémák az összes Soft Updates partíciót veszélyeztetik. Mennyiben vonatkoznak viszont ezek a gyökérpartícióra?

A gyökérpartíción nagyon ritkán változnak fontos információk. A /boot/kernel/kernel és az /etc egyedül a rendszer karbantartása során frissül, vagy például amikor a felhasználók jelszót változtatnak. Ha a rendszer egy ilyen változtatás harminc másodperces idején belül omlik össze,

akkor megvan rá az esélyünk, hogy elvesznek az adataink. Ez a kockázat a legtöbb alkalmazás számára elfogadható, de semmiképpen sem szabad figyelmen kívül hagynunk. Ha a rendszerünk nem képes vállalni még ennyi kockázatot sem, akkor a rendszerindító partíción tiltsuk le a Soft Updates használatát!

A gyökérpartíció hagyományosan az egyik legkisebb partíció. Ha viszont az ideiglenes állományok tárolására szánt /tmp könyvtárat is ezen belülre tesszük és gyakran használjuk, akkor ebből időszakosan tárhelyproblémáink adódhatnak. Könnyen megoldhatjuk azonban ezt a problémát, ha a /tmp könyvtárhoz létrehoznunk egy szimbolikus linket a /var/tmp könyvtárra.

9.5. Mi történt a ccd(4) eszközzel?

A hibajelenség:

```
# ccdconfig -C
ccdconfig: ioctl (CCDIOCSET): /dev/ccd0c: Inappropriate file type or format
```

Ez általában olyankor történik, amikor olyan `c` partíciókat próbálunk meg összefűzni, amelyek alapértelmezés szerint `unused` ("nem használt") típusúak. A ccd(4) meghajtó azonban megköveteli, hogy az érintett partíciók `FS_BSDFFS` típusúak legyenek. Szerkesszük át a lemezeken található címkéket és változtassuk meg a partíciók típusát a `4.2BSD` értékre.

9.6. Miért nem lehet a ccd(4) eszköz lemezcímkéjét szerkeszteni?

A hibajelenség:

```
# disklabel ccd0
(itt valami gondot ír ki, ezért megpróbáljuk szerkeszteni a címkét)
# disklabel -e ccd0
(edit, save, quit)
disklabel: ioctl DIOCWDINFO: No disk label on disk;
use "disklabel -r" to install initial label
```

Ezt általában azért kapjuk, mert a ccd(4) által visszaadott lemezcímke valójában "nem létezik" a lemezen. Ezen úgy tudunk segíteni, ha explicit módon visszaírjuk, valahogy így:

```
# disklabel ccd0 > /tmp/lemezcimke.tmp
# disklabel -Rr ccd0 /tmp/lemezcimke.tmp
# disklabel -e ccd0
(most már működni fog)
```

9.7. Lehet más operációs rendszerek állományrendszerét is csatlakoztatni FreeBSD alatt?

A FreeBSD több más állományrendszert is ismer.

UFS

Az UFS formátumú CD-k FreeBSD alatt közvetlenül csatlakoztathatóak. A Digital UNIX és más rendszerek UFS partícióit nem már annyira könnyû csatlakoztatni, ez leginkább a kérdéses operációs rendszer partícionálási megoldásaitól függ.

ext2/ext3

A FreeBSD támogatja az `ext2fs` és `ext3fs` partíciókat. Errõl bõvebben lásd a mount_ext2fs(8) man oldalt.

NTFS

A FreeBSD csak olvasni képes az NTFS partíciókat. Ezzel kapcsolatban a mount_ntfs(8) man oldalán találunk részletesebb információkat. Az írhatóság használatához az ntfs-3g portolt változatát javasoljuk (lásd sysutils/fusefs-ntfs).

FAT

A FreeBSD egyaránt képes írni és olvasni a FAT típusú partíciókat. Errõl a mount_msdosfs(8) man oldalán tudhatunk meg többet.

ReiserFS

A FreeBSD tudja olvasni a ReiserFS partíciókat. Ezt a mount_reiserfs(8) man oldalon olvashatjuk.

ZFS

A FreeBSD jelen pillanatban a Sun™ ZFS meghajtójának átiratát is tartalmazza. Jelenleg azonban csak elegendõ memóriával rendelkezõ amd64 platformokon javasoljuk a használatát. Részletesebb információkért lásd a zfs(8) man oldalt.

A FreeBSD hálózati állományrendszereket is támogat, többek közt az NFS-t (lásd mount_nfs(8)), a NetWare-t (lásd mount_nwfs(8)), és Microsoft-féle SMB állományrendszereket (lásd mount_smbfs(8)). Más egyéb FUSE-alapú állományrendszer (sysutils/fusefs-kmod) támogatását is megtalálhatjuk a portok között.

9.8. Hogyan lehet másodlagos (logikai) DOS partíciókat csatlakoztatni?

A logikai DOS partíciók az elsõdleges partíciók *után* találhatóak. Például, ha van egy "E" betûjelû logikai partíciónk a második SCSI-meghajtónkon, akkor lennie kell egy "ötödik slice-nak" a /dev könyvtárban, amelyet majd csatlakoztatni tudunk:

```
# mount -t msdosfs /dev/da1s5 /dos/e
```

9.9. Használható titkosított állományrendszer FreeBSD alatt?

Igen. Erre a célra a gbde(8) és a geli(8) is tökéletesen alkalmas. A részleteket lásd a FreeBSD kézikönyv A lemezpartíciók titkosítása címû fejezetében.

9.10. A Windows NT® rendszertöltõjével is el lehet indítani a FreeBSD-t?

Ehhez tulajdonképpen csak annyit kell csinálnunk, hogy átmásoljuk a FreeBSD rendszerindító partíciójának az elsõ szektorát egy állományba a DOS/Windows NT® partíción belül. Legyen ez például a C:\BOOTSECT.BSD állomány (a C:\BOOTSECT.DOS mintájára), amelyhez aztán így igazítjuk a c:\boot.ini állományt:

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Windows NT"
C:\BOOTSECT.BSD="FreeBSD"
C:\="DOS"
```

Ha a FreeBSD ugyanazon a lemezen található, ahonnan a Windows NT® is indul, akkor egyszerûen csak másoljuk át a /boot/boot1 állományt C:\BOOTSECT.BSD néven. Ha viszont a FreeBSD egy másik lemezen található, akkor a /boot/boot1 önmagában már nem elegendõ, hanem helyette a /boot/boot0 állományra lesz szükségünk.

A /boot/boot0 állományt a sysinstall(8) használatával kell telepíteni abból a menübõl, ahol a FreeBSD boot managerét kell kiválasztani. Erre azért van szükség, mert a /boot/boot0 állományon belül a partíciós tábla teljesen üres, azonban a sysinstall(8) át fogja másolni a partíciós táblát mielõtt a /boot/boot0 állományt az MBR-be tenné.

Ne másoljunk át csak úgy egyszerûen a /boot/boot0 állományt a /boot/boot1 helyett! Ezzel felülíródik a partíciós táblánk és így a számítógépet nem tudjuk elindítani!

Amikor a FreeBSD boot managere lefut, az utoljára indított operációs rendszert a partíciós táblában aktívként jelöli meg (ezzel lényegében megjegyzi), majd ezután beírja magát az MBR-be. Emiatt, hogy ha csak egyszerûen átmásoljuk a /boot/boot0 állományt a C:\BOOTSECT.BSD állományba, akkor csak egy egyetlen aktív bejegyzést tartalmazó üres partíciós táblát fog visszaírni az MBR-be.

9.11. A LILO-ból hogyan lehet FreeBSD-t és Linuxot is indítani?

Ha a FreeBSD és a Linux® is ugyanazon a lemezen helyezkedik el, akkor nincs más teendõnk, mint követni a LILO telepítési útmutatójában a nem-Linux® típusú operációs rendszerek indítására

vonatkozó utasításokat. Ezek röviden összefoglalva a következõk:

Indítsuk el a Linuxot és vegyük fel a következõ sort az /etc/lilo.conf állományba:

```
other=/dev/hda2
        table=/dev/hda
        label=FreeBSD
```

(A fentiekben feltételeztük, hogy a FreeBSD-t tartalmazó slice a Linux® számára /dev/hda2 néven érhetõ el. Ez természetesen a saját konfigurációnkhoz kell szabni.) Ezután egyszerûen csak futtassuk le a `lilo` parancsot `root` felhasználóként és már készen is vagyunk.

Ha a FreeBSD egy másik lemezen található, akkor a `loader=/boot/chain.b` LILO-bejegyzést kell használnunk. Például:

```
other=/dev/dab4
        table=/dev/dab
        loader=/boot/chain.b
        label=FreeBSD
```

Bizonyos helyzetekben elõfordulhat, hogy a FreeBSD rendszertöltõjének át kell adnunk a meghajtó BIOS szerinti sorszámát, mert csak így tudjuk rendesen elindítani a második lemezrõl. Például, ha a FreeBSD szerint a SCSI-lemezünk a BIOS-ban az 1-es lemez, akkor ezt kell megadnunk a FreeBSD rendszertöltõjének:

```
Boot: 1:da(0,a)/boot/kernel/kernel
```

A boot(8) beállítható úgy, hogy a rendszer indításakor automatikusan mindig ezt a beállítást használja.

A FreeBSD és Linux® együttes használatáról további részleteket a Linux®+FreeBSD mini-HOWTO címû írásból tudhatunk meg.

9.12. Hogyan lehet a GRUB használatával FreeBSD-t és Linuxot is indítani?

A FreeBSD-t nagyon könnyû elindítani a GRUB segítségével. Ehhez csupán annyit kell tennünk, hogy felvesszük a következõ sorokat a GRUB konfigurációs állományába (/boot/grub/menu.lst, vagy bizonyos, például Red Hat-típusú rendszerekben a /boot/grub/grub.conf):

```
title FreeBSD 6.1
    root (hd0,a)
    kernel /boot/loader
```

Itt a *hd0,a* az elsõ lemezen található rendszerindító partícióra mutat. Ha a lemezen belül a slice számát is szeretnénk megadni, akkor írhatjuk így is: *(hd0,2,a)*. Ha ezt nem adjuk meg, akkor a GRUB alapértelmezés szerint a lemezen levõ elsõ a partícióval rendelkezõ slice-ot keresi meg.

9.13. Hogyan lehet a BootEasy használatával elindítani a FreeBSD-t és a Linuxot?

A LILO-t ne a Master Boot Recordba, hanem a linuxos partíciónk elejére telepítsük. Ezután a BootEasybõl már el tudjuk indítani a LILO-t.

Abban az esetben is ezt javasoljuk, ha Windows® és Linux® is van a gépünkön, mivel így szintén egyszerûbb lesz elindítani a Linuxot, ha netalán valamikor újra kellene telepíteni a Windows®-t (ami viszont egy "irigy" operációs rendszer, mert nem tûr meg semmilyen más operációs rendszert maga mellett a Master Boot Recordban).

9.14. A rendszerindításkor látható ??? hogyan írható át valami értelmesre?

Ez az szabványos boot managerrel csak úgy lehet megoldani, ha újratelepítjük. A portok között viszont a sysutils kategóriában rengeteg olyan más boot managert találhatunk, amely tud ilyet is.

9.15. Cserélhetõ lemezes meghajtókat hogyan lehet használni?

Legyen az akár egy Zip®, EZ drive meghajtó (esetleg egy floppy, ha így akarjuk használni), vagy éppen egy új merevlemez, miután már telepítettük és felismerte a rendszert, illetve behelyeztük a lemezt, kártyát vagy akármit, minden esetben szinte ugyanaz a teendõ.

(Ez a válasz leginkább Mark Mayo ZIP GYIK címû írásán alapszik.)

Ha tehát egy ZIP meghajtóról vagy floppylemezrõl beszélünk, amelyen egy DOS-os állományrendszer található, akkor azt parancssorból így érhetjük el, ha floppy:

```
# mount -t msdosfs /dev/fd0c /floppy
```

vagy így, ha egy gyári beállításokkal rendelkezõ ZIP-lemez:

```
# mount -t msdosfs /dev/da2s4 /zip
```

A többi lemez esetén a fdisk(8) vagy a sysinstall(8) segítségével nézzük meg, hogy milyen partíciók és hogyan találhatóak meg rajtuk.

A következõ példákban egy da2 eszközként, vagyis egy harmadik SCSI-lemezként megjelenõ ZIP-meghajtót fogunk használni.

Hacsak nem floppyval van dolgunk, illetve nem tervezzük másoknak is odaadni a cserélhetõ médiumot, akkor érdemes inkább BSD típusú állományrendszert telepíteni rá. Így támogatottak lesznek a hosszú állománynevek, és legalább egy kétszer gyorsabb és egy sokkal megbízhatóbb megoldást kapunk. Ehhez elõször is le kell szednünk a DOS-szintû partíciókat és állományrendszereket. Erre a célra egyaránt megfelel a fdisk(8) vagy a sysinstall(8), illetve kisebb lemezek esetén valószínûleg nem is lesz szükségünk több operációs rendszer támogatására, így aztán közvetlenül is törülhetjük ezeket:

```
# dd if=/dev/zero of=/dev/rda2 count=2
# disklabel -Brw da2 auto
```

A disklabel(8) vagy a sysinstall(8) használatával ezután létre tudunk hozni BSD típusú partíciókat. Valószínûleg erre lesz szükségünk, ha lapozóállományt is tenni akarunk a lemezre, noha ennek nem sok értelme van például egy ZIP-meghajtó esetén.

Végezetül hozzunk létre egy új állományrendszert. Itt most ez egész ZIP-lemezen egyetlen partíció lesz:

```
# newfs /dev/rda2c
```

Csatlakoztassuk:

```
# mount /dev/da2c /zip
```

Emellett még hasznos lehet felvenni hozzá egy sort az /etc/fstab állományba is (lásd fstab(5)), így a jövõben elegendõ csak a `mount /zip` parancsot kiadnunk a csatlakoztatásához:

```
/dev/da2c /zip ffs rw,noauto 0 0
```

9.16. Miért ad a rendszer Incorrect super block hibát CD-k csatlakoztatásánál?

Fel kell világosítanunk a mount(8) parancsot a csatlakoztatandó eszköz típusáról. Errõl a kézikönyv lézeres tárolóeszközökrõl szóló részében olvashatunk, innen is különösen a Adat CD-k használata címû szakaszt ajánljuk.

9.17. Miért ad a rendszer Device not configured hibaüzenetet CD-k csatlakoztatásakor?

Ez általában arra utal, hogy nincs CD a meghajtóban, vagy a meghajtó nem érhetõ el a buszon. Ezzel kapcsolatban a kézikönyv Adat CD-k használata címû szakaszát javasoljuk elolvasásra.

9.18. Miért jelenik meg az összes nemzeti karakter helyén ?, amikor FreeBSD alatt csatlakoztatunk egy CD-t?

A CD-n valószínûleg a "Joliet" kiterjesztés használatával tárolják az állományok és könyvtárak adatait. Erre vonatkozóan a kézikönyvben a Lézeres tárolóeszközök (CD-k) létrehozása és használata címû rész elolvasását javasoljuk, különös tekintettel az Adat CD-k használata címû szakaszra.

9.19. A FreeBSD alatt készített CD-ket nem lehet más operációs rendszerekkel olvasni. Miért nem?

Ez minden bizonnyal abból fakad, hogy nem egy ISO 9660 állományrendszert vettük fel rá, hanem közvetlenül maguk az állományokat. Olvassuk el a kézikönyvben a Lézeres tárolóeszközök (CD-k) létrehozása és használata címû fejezetet, de különösen a Nyers adat CD-k írása címû részt.

9.20. Hogyan lehet lementeni egy adat CD tartalmát a merevlemezre?

Errõl a kézikönyvben találunk hasznos információkat, azon belül is az Adat CD-k másolása címû szakaszban. A CD-kkel végezhetõ további mûveletekrõl a kézikönyv Lézeres tárolóeszközök (CD-k) létrehozása és használata címû részében találhatunk részletes útmutatásokat.

9.21. Miért nem lehet audio CD-ket csatlakoztatni a mount paranccsal?

Ha zenei CD-ket próbálunk meg csatlakoztatni, akkor például egy `cd9660: /dev/acd0c: Invalid argument` hibát fogunk kapni a rendszertõl. Ez azért történik, mert a `mount` parancs csak állományrendszerekkel használható. A zenei CD-ken viszont semmilyen állományrendszer nincs, egyszerûen csak maga az adat. Az olvasásukhoz olyan programra lesz szükségünk, amely képes zenei CD-kkel dolgozni, mint például az audio/xmcd port.

9.22. Hogyan lehet többmenetes (multisession) CD-ket csatlakoztatni a mount paranccsal?

A mount(8) alapértelmezés szerint az CD-n található utolsó adatsávot (menetet, vagy sessiont) próbálja meg olvasni. Ha viszont egy korábbi menetet szeretnénk vele betöltetni, akkor erre használjuk a `-s` paranccsori paramétert. Erre a mount_cd9660(8) man oldalon találhatunk különbözõ példákat.

9.23. Hogyan képesek az egyszerû felhasználók floppykat, CD-ket és más egyéb cserélhetõ lemezes eszközöket használni?

A normál felhasználók számára engedélyezni tudjuk az eszközök csatlakoztatását. Íme:

1. `root` felhasználóként állítsuk be a `vfs.usermount` sysctl változót az `1` értékre:

   ```
   # sysctl -w vfs.usermount=1
   ```

2. A cserélhetõ eszközöket képviselõ eszközleírókra állítsuk be `root` felhasználóként a megfelelõ engedélyeket.

   Például a felhasználóknak így tudjuk engedélyezni az elsõ floppymeghajtó használatát:

   ```
   # chmod 666 /dev/fd0
   ```

   Az `operator` csoportban levõ felhasználók pedig így fognak tudni CD-ket csatlakoztatni:

   ```
   # chgrp operator /dev/acd0c
   # chmod 640 /dev/acd0c
   ```

3. Fel kell vennünk ezeket a módosításokat az /etc/devfs.conf állományba is, mivel csak így maradnak meg a következõ rendszerindítás után.

   Ehhez `root` felhasználóként a vegyük fel a megfelelõ sorokat az /etc/devfs.conf állományba. Például, ha a felhasználóknak engedélyezni akarjuk az elsõ floppymeghajtó használatát, akkor:

   ```
   # Bármelyik felhasználó képes floppykat csatlakoztatni.
   own       /dev/fd0    root:operator
   perm      /dev/fd0    0666
   ```

   Így engedélyezhetjük az `operator` csoport tagjainak a CD-k csatlakoztatását:

   ```
   # Az operator csoport tagjai csatlakoztathatnak CD-ket.
   own       /dev/acd0   root:operator
   perm      /dev/acd0   0660
   ```

4. Végezetül tegyük a `vfs.usermount=1` sort az /etc/sysctl.conf állományba, így a rendszer következõ indításakor is megmarad ez a beállítás.

Most már mindegyik felhasználó képes csatlakoztatni a /dev/fd0 eszközleírón keresztül elérhetõ lemezt a saját könyvtárába:

```
% mkdir ~/az-én-csatlakozási-pontom
% mount -t msdosfs /dev/fd0 ~/az-én-csatlakozási-pontom
```

A `operator` csoport tagjai is képesek most már az /dev/acd0c eszközleírón keresztül elérhetõ CD-ket csatlakoztatni a saját könyvtárukba:

```
% mkdir ~/az-én-csatlakozási-pontom
% mount -t cd9660 /dev/acd0c ~/az-én-csatlakozási-pontom
```

Az eszközök leválasztása is hasonlóan egyszerû:

```
% umount ~/az-én-csatlakozási-pontom
```

A `vfs.usermount` engedélyezésével azonban együttjár némi biztonsági kockázat is. Az MS-DOS® formátumú lemezek csatlakoztatására ezért inkább a Portgyûjteményben található emulators/mtools csomagot javasoljuk.

A példákban használt eszközneveket természetesen a konfigurációnknak megfelelõen meg kell változtatnunk.

9.24. A du és a df parancsok eltérõ mennyiségû szabad helyet mutatnak. Mi okozza ezt?

A válaszhoz meg kell értenünk a `du` és a `df` mûködését. A `du` végigmegy a könyvtárszerkezeten és megnézi, hogy mekkorák az egyes állományok, majd megjeleníti a végösszegüket. A `df` ezzel szemben egyszerûen csak lekérdezi az állományrendszertõl, hogy mennyi szabad hely maradt rajta. Ezek látszólag ugyanazt a módszer fedik, azonban miközben a könyvtár nélkül állományok befolyásolják a `df` parancsot, addig a `du` parancsot nem.

Amikor egy program használ egy olyan állományt, amelyet eközben letörlünk, egészen addig létezni fog, amíg a program be nem fejezi a használatát. Ettõl függetlenül viszont az állomány azonnal eltûnik a könyvtárból. Ezt nagyon könnyen ki is tudjuk próbálni egy olyan programmal, mint például a `more`. Tegyük fel, hogy van akkora állományunk, amely elég nagy ahhoz, hogy feltûnjön a `du` és a `df` kimenetében. (Mivel manapság már nagyok a tárolóeszközök, ennek egy *igen nagy* állománynak kell lennie!) Ha letöröljük ezt az állományt, miközben a `more` paranccsal még használjuk, a `more` nem fog rögtön leállni és panaszkodni az állomány hiányára. Egyedül csak az állományhoz tartozó bejegyzés tûnik el a könyvtárból, így más program már nem tud hozzáférni. A `du` erre már azt mondja, hogy nem létezik - bejárta a könyvtárat és nem találta. A `df` szerint azonban még mindig ott van, hiszen az állományrendszer tudja, hogy a `more` parancsnak még szüksége van rá. Ahogy a `more` befejezte a dolgát, a `du` és a `df` által mutatott értékek ismét egyezni fognak.

Azt sem szabad elfelejtenünk, hogy a Soft Updates használata esetén akár 30 másodpercet is

várnunk kell, hogy a változtatásaink láthatóvá váljanak!

Ez a helyzet nagyon gyakori webszerverek esetén. Sokan úgy állítanak be a FreeBSD rendszerükön webszervert, hogy elfelejtik beállítani hozzá a naplók archiválását és váltását. Ilyenkor a hozzáférések naplózása gyorsan meg tudja tölteni a /var könyvtárat. Ekkor a rendszergazda törli az adott állományt, de a rendszer még mindig panaszkodik a szabad hely hiánya miatt. A webszerver leállítása és újraindítása ekkor segít felszabadítani az állományt, így az állományrendszerről is törlődhet. Ennek megelőzésére használjuk a newsyslog(8) programot.

9.25. Hogyan lehet növelni a lapozóterületet?

A kézikönyv Beállítás és finomhangolás című fejezetében található egyik szakaszban olvashatunk erről.

9.26. A FreeBSD miért látja kisebbnek a lemezeket mint amekkorának a gyártó mondja ezeket?

A merevlemezek gyártói általában a gigabyte-okat egy milliárd byte-ként számolják, miközben a FreeBSD pedig 1 073 741 824 byte-nak. Ez remekül megmagyarázza, hogy a FreeBSD rendszerüzenetei között egy elméletileg 80 GB méretű lemez miért 76 319 MB-osnak jelenik meg.

Emellett érdemes még tisztában lennünk azzal is, hogy a FreeBSD (alapértelmezés szerint) fenntartja a lemezterület 8 százalékát.

9.27. Hogyan lehet egy partíció 100 százaléknál is jobban megtelt?

Az UFS partíciók egy részét (amely alapértelmezés szerint a teljes kapacitás 8 százaléka) az operációs rendszer fenntartja a saját és a `root` felhasználó számára. A df(1) ezt a területet nem számolja a `Capacity` oszlopban megjelenő értékhez, ezért tudja átlépni a 100 százalékos arányt. Sőt még azt is láthatjuk, hogy a blokkok számát jelző `Blocks` oszlopban megjelenő érték mindig, általában pontosan 8 százalékkal nagyobb, mint a használt blokkokat jelző `Used` és a rendelkezésre álló blokkokat jelző `Avail` oszlopokban szereplő értékek összege.

A részleteket a tunefs(8) man oldalon belül a `-m` opció bemutatásánál olvashatjuk.

Chapter 10. Rendszeradminisztráció

10.1. Hol vannak a rendszerindítás beállításáért felelõs állományok?

Az ezzel kapcsolatos beállítások elsõsorban az /etc/defaults/rc.conf állományban találhatóak (lásd rc.conf(5)). A rendszer indításáért felelõs szkriptek, mint például az /etc/rc vagy az /etc/rc.d könyvtár tartalma (lásd rc(8)) ezt használja. *Ezt az állományt tilos közvetlenül szerkeszteni!* Ha valamit meg akarunk változtatni az /etc/defaults/rc.conf állományban szereplõ beállítások közül, akkor ehelyett egyszerûen csak másoljuk le az /etc/rc.conf állományba és állítsuk be ott az értékét.

Például, ha el akarjuk indítani a beépített névfeloldó szolgáltatást, a named(8) démont, akkor ennyit kell tennünk:

```
# echo 'named_enable="YES"' >> /etc/rc.conf
```

Ha helyi szolgáltatásokat akarunk futtatni, akkor tegyük a hozzá tartozó szkripteket az /usr/local/etc/rc.d könyvtárba. Ezek a szkriptek legyenek végrehajthatóak és az alapértelmezett állománymóduk legyen 555.

10.2. Hogyan lehet felhasználókat egyszerûen létrehozni?

Használjuk a adduser(8), vagy bonyolultabb esetekben a pw(8) parancsot.

Felhasználókat törölni a rmuser(8), vagy amennyiben szükséges, a pw(8) paranccsal tudunk.

10.3. A crontab szerkesztése után miért jelennek meg a root: not found és a hozzá hasonló hibaüzenetek?

Ilyen általában olyankor történik, amikor a rendszerszintû crontab állományt módosítjuk (/etc/crontab), majd a crontab(1) használatával megpróbáljuk telepíteni:

```
# crontab /etc/crontab
```

Ezt nem így kell megoldani. A rendszerszintû crontab felépítése eltér a felhasználókhoz tartozó crontab állományokétól (a crontab(5) man oldal szemlélteti részletesebben ezeket az eltéréseket), amelyet a crontab(1) próbál meg ilyenkor telepíteni.

Ha így csináltuk, akkor a crontab nem lesz több, mint az /etc/crontab hibás formátumú változata. Töröljük le:

```
# crontab -r
```

Legközelebb, amikor az /etc/crontab állományt módosítjuk, nem kell értesítenünk a cron(8) démont, mivel magától észre fogja venni az elvégzett változtatásokat.

Ha valamit napi, heti vagy havi rendszerességgel akarunk futtatni, akkor ehelyett inkább másoljuk be az /usr/local/etc/periodic könyvtárba, és hagyjuk, hogy a `cron` hívja meg a periodic(8) parancson keresztül az összes többi rendszeresen elvégzendõ feladattal együtt.

Ez a hiba egyébként onnan jön, hogy rendszerszintû crontab állomány esetén van még egy további mezõ, amely megadja, hogy az adott parancsot melyik felhasználóval kell futtatni. Az alapértelmezett rendszerszintû crontab állomány esetén ez mindenhol a `root`. Amikor ezt a crontab állományt a `root`crontab állományaként használjuk (amely *nem* ugyanaz, mint a rendszerszintû crontab), akkor a cron(8) a `root` szót a végrehajtandó parancs részének fogja tekinteni, amely viszont nem létezik.

10.4. Miért jelenik meg a you are not in the correct group to su root hibaüzenet, amikor a su paranccsal át akarunk váltani a root felhasználóra?

Ez egy biztonsági megszorítás. Csak úgy tudunk átváltani a `root` felhasználóra (vagy bármilyen más olyan hozzáférésre, amely rendszeradminisztrátori jogosultságokkal rendelkezik), ha a `wheel` csoport tagjai vagyunk. Ha nem létezne ez a korlátozás, akkor a rendszerben szinte bárki képes lenne rendszeradminisztrátori jogosultságokat szerezni csupán úgy, hogy ha megszerzi valahogy a `root` jelszavát. Ennek a korlátozásnak köszönhetõen ez viszont már nem lesz feltétlenül helytálló. A su(1) még a jelszót sem engedi megadni azoknak, akik nem tagjai a `wheel` csoportnak.

Ha engedélyezni akarjuk valakinek a `root` felhasználóra váltást, akkor nincs más teendõnk, mint egyszerûen a hozzáadni a `wheel` csoporthoz.

10.5. Az rc.conf állományban vagy valamelyik másik konfigurációs állományban rosszul adtuk meg a beállításokat, és nem lehet módosítani ezeket, mert így írásvédett lett az állományrendszer. Mi a megoldás?

Indítsuk újra a rendszert és a rendszertöltõ parancssorában adjuk ki a `boot -s` parancsot, amivel így egyfelhasználós módba váltunk. Amikor meg kell adnunk a használni kívánt parancsértelmezõ nevét, egyszerûen csak nyomjuk le az `Enter` billentyût, majd a `mount -urw /` parancs kiadásával csatlakoztassuk újra írható módban rendszerindító állományrendszert. Emellett még valószínûleg a `mount -a -t ufs` paranccsal azokat az állományrendszereket is érdemes lesz csatlakoztatnunk, ahol a kedvenc szövegszerkesztõnk található. Amennyiben az érintett szövegszerkesztõ egy hálózati állományrendszeren található, akkor helyette használjunk egy helyben elérhetõ szövegszerkesztõt,

például az ed(1) programot, vagy manuálisan állítsuk be a hálózat elérését a hálózati állományrendszerek csatlakoztatásához.

Ha a vi(1) vagy emacs(1) programokhoz hasonló teljes képernyõs szövegszerkesztõt akarunk használni, akkor elõtte nem árt a `export TERM=cons25` parancsot sem kiadnunk, így a termcap(5) adatbázisból elérhetõvé válnak az ehhez szükséges adatok.

Miután megtettük ezeket a lépéseket, már a szokásos módon át tudjuk szerkeszteni az /etc/rc.conf állományt. A rendszermag indulása után közvetlenül megjelenõ üzenetekben találhatjuk meg azon sorok számait, amelyeket a rendszer nem tudott értelmezni.

10.6. Miért nem sikerül beállítani a nyomtatót?

Olvassuk el a kézikönyv nyomtatókkal foglalkozó részét, minden bizonnyal választ ad a legtöbb kérdésünkre.

Bizonyos nyomtatókat azonban akkor tudunk használni, ha van hozzá meghajtónk. Ezeket gyakran csak "WinPrinter" néven emlegetik, amelyeket viszont a FreeBSD nem támogat. Ha a nyomtatónk nem használható DOS vagy Windows® alatt, akkor valószínûleg egy ilyen WinPrinterrel van dolgunk. Ebben az esetben egyedül abban reménykedhetünk, hogy a print/pnm2ppa port támogatja.

10.7. Hogyan lehet módosítani a rendszerünkhöz tartozó billentyûkiosztást?

Olvassuk el a kézikönyv honosításssal foglalkozó részét, különös tekintettel a konzol beállításaira.

10.8. Miért jelenik meg az unknown: <PNP0303> can’t assign resources hibaüzenet a rendszer indulásakor?

Erre a FreeBSD-CURRENT levelezési lista címére postázott egyik levél adja meg a választ:

Garrett Wollman <wollman@FreeBSD.org>, 2001. április 24. A "can’t assign resources" üzenetek rendszerünkben olyan ISA eszközök jelenlétére utalnak, amelyekhez a rendszermagban PnP támogatást nem tartalmazó meghajtók tartoznak. Ilyenek többek közt a billentyûzetvezérlõk, a programozható megszakítás-vezérlõ chip és sok más alapvetõ elem a gépünkben. Ezek az erõforrások nem oszthatóak ki, mivel már valamelyik meghajtó használatba vette ezeket.

10.9. Miért nem mûködnek rendesen a kvóták?

1. Elõfordulhat, hogy a rendszermag nem támogatja a kvóták használatát. Ha errõl lenne szó, akkor vegyük fel az alábbi sort a rendszermag konfigurációs állományába és fordítsuk újra:

   ```
   options QUOTA
   ```

Ennek részleteit akézikönyv kvótákkal foglalkozó részében találjuk.

2. Az / állományrendszeren ne engedélyezzük a kvóták használatát.
3. Tegyünk kvótaállományokat azokra az állományrendszerekre, ahol be akarjuk vezetni a használatukat, például:

Állományrendszer	Kvótaállomány
/usr	/usr/admin/quotas
/home	/home/admin/quotas
...	...

10.10. A FreeBSD tartalmazza a System V IPC alapeszközeit?

Igen, a FreeBSD a GENERIC típusú rendszermagban támogatja a System V típusú IPC megoldást, beleértve az osztott memória, az üzenetek és a szemaforok használatát. Ha saját rendszermagunk van, akkor az alábbi beállítások használatával engedélyezhetjük a használatukat:

```
options    SYSVSHM          # az osztott memória engedélyezése
options    SYSVSEM          # a szemaforok engedélyeze
options    SYSVMSG          # az üzenetek kezelése
```

Fordítsuk és telepítsük újra a rendszermagot.

10.11. A sendmail helyett milyen más levelezõ szerver használható még?

A sendmail a FreeBSD-ben található alapértelmezett levelezõ szerver, de könnyen le tudjuk cserélni másikra (például amelyet a portok közül telepítettünk).

A Portgyûjteményben több különbözõ levelezõ szerver is megtalálható, amelyek közül a mail/exim, mail/postfix, mail/qmail és a mail/zmailer portok a leginkább népszerûek.

Szép dolog, hogy lehet válogatni a különbözõ megoldások között és hogy ilyen sok levelezõ szerver használható. Ezért lehetõleg a levelezési listákon ne kérdezzünk senkitõl olyat, hogy "De a sendmail akkor most miért jobb, mint a qmail?" Ha ilyen kérdéseink vannak, akkor elõször inkább olvassuk át az archívumokat. Szinte biztos, hogy már szinte az összes levelezõ szerver elõnyét és hátrányát kivesézték jó néhányszor.

10.12. Elveszett a root felhasználó jelszava! Mit tegyünk?

Ne essünk kétségbe! Indítsuk újra a rendszerünket egyfelhasználós módban. Ehhez gépeljük be a

`boot -s` parancsot a rendszertöltõ `Boot:` parancssorában. Amikor a parancsértelmezõt kell megadnunk, egyszerûen csak nyomjuk le az `Enter` billentyût. Ekkor kapunk egy # parancssort. A `mount -urw /` parancs begépelésével csatlakoztassuk újra a rendszerindító partíciónkat írható módban, majd a `mount -a` paranccsal csatlakoztassuk az összes többi állományrendszert. Ezt követõen a `passwd root` parancs kiadásával változtassuk meg a `root` felhasználó jelszavát és a exit(1) futtatásával folytassuk a rendszer indítását.

Ha az egyfelhasználós módra váltás során a rendszer a `root` felhasználó jelszavát kérné, akkor az arra utal, hogy a konzol (/dev/console) az /etc/ttys állomány szerint `insecure` (nem biztonságos) típusú. Ebben az esetben szereznünk kell egy FreeBSD telepítõlemezt, elindítanunk róla a rendszert, majd a sysinstall(8) programban a Fixit menüponton keresztül indított parancsértelmezõben kiadni az elõbb említett parancsokat.

Ha egyfelhasználós módban nem tudjuk csatlakoztatni a rendszerindító partíciót, akkor ennek könnyen az lehet az oka, hogy a partíciókat titkosították, ezért a megfelelõ kulcsok nélkül nem tudjuk elérni ezeket. Ez leginkább adott implementációtól függ. A FreeBSD-ben elõforduló lemeztitkosításokkal kapcsolatban a kézikönyv ad bõvebb útmutatást.

10.13. Hogyan akadályozható meg, hogy a ControlAltDelete billentyûkombináció újraindítsa a rendszert?

Ha a syscons(4) (vagyis az alapértelmezett) konzolt használjuk, akkor ehhez a következõ beállításokkal kell fordítanunk és telepítenünk egy rendszermagot:

```
options SC_DISABLE_REBOOT
```

Mindezt a rendszermag újrafordítása és a újraindítása nélkül is le tudjuk tiltani, ha beállítjuk az alábbi sysctl(8)-változót:

```
# sysctl hw.syscons.kbd_reboot=0
```

Az elõbb említett két módszer kizárja egymást. A sysctl(8) változó nem létezik, ha a rendszermagot a `SC_DISABLE_REBOOT` beállítással fordítjuk újra.

Ha viszont a pcvt(4) konzolt használjuk, akkor a következõ konfigurációs beállítást kell megadnunk a rendszermag újrafordításakor:

```
options PCVT_CTRL_ALT_DEL
```

10.14. Hogyan lehet szöveges DOS állományokat UNIX® formátumúra alakítani?

Használjuk a következõ perl(1) parancsot:

```
% perl -i.bak -npe 's/\r\n/\n/g' állományok
```

ahol az *állományok* az átalakítandó állományok. A konverzió helyben történik, illetve az eredeti állományokról .bak kiterjesztéssel létrejön egy biztonsági mentés.

Erre a célra viszont ugyanígy megfelel a tr(1) parancs is:

```
% tr -d '\r' < dos-szöveges-állomány > unix-szöveges-állomány
```

Ekkor a *dos-szöveges-állomány* lesz a DOS formátumú szöveges állomány, miközben a *unix-szöveges-állomány* fogja az eredményt tartalmazni. Ez valamivel gyorsabb a `perl` megoldásánál.

Ez említett megoldásokon kívül a DOS szöveges állományait a Portgyûjteményben található converters/dosunix porttal is könnyedén át tudjuk alakítani. Ennek részleteit a hozzá tartozó dokumentációból tudjuk meg.

10.15. Hogyan lehet futó programokat név szerint leállítani?

Lásd killall(1).

10.16. A su(1) miért írja folyton, hogy a felhasználó nincs a root ACL-jében?

Ezt a hibát az elosztott hitelesítést végzõ Kerberos rendszer adja. Maga a probléma nem végzetes, viszont annál inkább idegesítõ. Ilyenkor vagy a `-K` kapcsolóval kell futtatni a su(1) programot, vagy a következõ kérdésben megadottak szerint el kell távolítani a Kerberos alkalmazást.

10.17. Hogyan távolítható el a Kerberos?

A Kerberos úgy távolítható el a rendszerbõl, ha újratelepítjük a `base` terjesztés tartalmát. Ha CD-rõl telepítettük a rendszert, akkor csatlakoztassuk (most tegyük fel, hogy a /cdrom könyvtárba) és futassuk a következõ parancsot:

```
# cd /cdrom/base
# ./install.sh
```

Másik lehetõség, ha hozzáadjuk a `NO_KERBEROS` beállítást a /etc/make.conf állományhoz és

újrafordítjuk az alaprendszert.

10.18. Mi történt a /dev/MAKEDEV állománnyal?

A FreeBSD 5.*X* és a későbbi változatok már a devfs(8) által felkínált automatikus megoldást alkalmazzák. Ilyenkor az eszközmeghajtók igény szerint hoznak létre eszközleírókat, és ezzel lényegében szükségtelenné teszik a /dev/MAKEDEV használatát.

10.19. Hogyan lehet még több pszeudoterminált létrehozni?

Ha sok `telnet`, `ssh`, X esetleg `screen` felhasználónk van, akkor könnyen elõfordulhat, hogy kifogyunk a pszeudoterminálokból. A FreeBSD 6.2 és az azt megelõzõ változatokban alapértelmezés szerint 256 pszeudoterminál, a FreeBSD 6.3 és késõbbi változatokban pedig 512 pszeudoterminál áll rendelkezésünkre.

Szükség esetén további pszeudoterminálok is hozzáadhatóak a rendszerhez. Ehhez azonban módosítanunk kell a szabványos C függvénykönyvtárakat, a rendszermagot és az /etc/ttys állományt. Például a http://www.freebsd.org/~jhb/patches/pty_1152.patch 1152 pszeudoterminál használatát teszi lehetõvé. Ez a konkrét javítás viszont csak a FreeBSD 6.3 és késõbbi változatok esetén alkalmazható zökkenõmentesen.

10.20. Hogyan lehet újraindítás nélkül az /etc/rc.conf tartalmát újraolvastatni és újraindítani az /etc/rc szkriptet?

Váltsunk egyfelhasználós módba, majd vissza többfelhasználós módba.

Konzolon ez így oldható meg:

```
# shutdown now
(Megjegyzés: nincs -r vagy -h!)

# return
# exit
```

10.21. A -STABLE rendszer frissítésekor -BETAx, -RC vagy -PRERELEASE verzió jelenik meg! Mi történt?

Röviden: Ez csak egy elnevezés. Az *RC* jelentése "Release Candidate", vagyis "kiadásra jelölt". Ez egy küszöbön álló kiadásra utal. A FreeBSD-ben a *-PRERELEASE* elnevezés általában egyenlõ a kiadások elõtt bekövetkezõ kódfagyasztással. (Bizonyos kiadások esetén pedig a *-BETA* címkét a

-PRERELEASE megjelöléshez hasonlóan használják.)

Valamivel bõvebben: A FreeBSD fejlesztésében a kiadások általában két helyrõl származnak. A nagyobb, ún. "nullás" kiadások, mint például 7.0-RELEASE és 8.0-RELEASE, a fejlesztési ág legfrissebb állapotából készülnek, amelyet gyakran csak -CURRENT néven emlegetnek. A kisebb kiadások, mint például a 6.3-RELEASE vagy az 5.2-RELEASE, az aktív -STABLE ágból származnak. A 4.3-RELEASE kiadástól kezdõdõen mindegyik kiadás saját ággal rendelkezik, amelyet elsõsorban olyanoknak ajánlunk, akiknek csak nagyon visszafogott változtatásokra van szükségük a rendszerben (ezek általában csak különbözõ biztonsági javításokat takarnak).

Amikor a fejlesztõk készíteni akarnak egy újabb kiadást, az alapjául szolgáló fejlesztési ágon elvégeznek bizonyos mûveleteket. Ennek egy része a források "befagyasztása". Amikor ez megkezdõdik, az ág neve megváltozik, és ezzel jelzik, hogy hamarosan kiadás készül belõle. Például, ha egy ág a 6.2-STABLE nevet viseli, akkor a 6.3-PRERELEASE névre vált arra az idõszakra, amíg tart a kódfagyasztás és lezajlik a kiadások megjelentetéséhez szükség további tesztelés. Hibajavítások ekkor továbbra is rakhatóak bele. Ahogy a források elérik a kiadáshoz szükséges szintet, az ág neve 6.3-RC-re vált, és ezzel jelzik, hogy a kiadás elõkészítése hamarosan befejezõdik. Az *RC* állapotban csak a legfontosabb hibákat keresik meg és javítják. Miután a kiadás (jelen esetünkben a 6.3-RELEASE kiadás) és a hozzá tartozó ág elkészült, az ág neve ismét 6.3-STABLE lesz.

A verziószámokról és a CVS-ben található különbözõ ágakról a Release Engineering címû cikkben olvashatunk (angolul).

10.22. Az új rendszermag telepítése során a chflags(1) program hibát jelez. Hogyan javítható ez a hiba?

Rövid válasz: A rendszerünk valószínûleg nullánál nagyobb biztonsági szinten fut. Indítsuk újra a rendszerünket egyfelhasználós módban és úgy telepítsük a rendszermagot.

A hosszabb válasz: A FreeBSD nem engedi megváltoztatni a rendszerszintû állományjelzõket nullától a nagyobb biztonsági szinteken. A jelenleg érvényben levõ biztonsági szintet a következõ paranccsal lehet lekérdezni:

```
# sysctl kern.securelevel
```

A biztonsági szintet nem lehet csökkenteni. A rendszert egyfelhasználós módban kell újraindítani, mert csak úgy tudjuk újratelepíteni a rendszermagot. Másik lehetõségünk, ha átállítjuk a biztonsági szintet az /etc/rc.conf állományban és úgy indítjuk újra a rendszerünket. Az init(8) man oldalán olvashatunk bõvebben a biztonsági szintek (`securelevel`) beállításáról, az rc.conf használatáról pedig az /etc/defaults/rc.conf állományból és a rc.conf(5) man oldalon tudhatunk meg többet.

10.23. A rendszeren nem lehet egyszerre egy másodpercnél többel megváltoztatni az idõt! Hogyan lehet megkerülni ezt a korlátozást?

A rövid válasz: A rendszerünkben a biztonsági szintet (`securelevel`) minden bizonnyal egynél nagyobbra állították. Indítsuk újra a rendszert egyfelhasználós módban és változtassuk meg a dátumot.

Egy hosszabb válasz: A FreeBSD nem engedi egy másodpercnél többel megváltoztatni az idõt, ha az aktuális biztonsági szint értéke egy felett van. Ezt a következõ parancs kiadásával tudjuk ellenõrizni:

```
# sysctl kern.securelevel
```

A biztonsági szint futás közben nem csökkenthetõ. A dátum megváltoztatásához ezért a rendszert egyfelhasználós módban kell indítanunk, vagy az /etc/rc.conf állományban csökkentenünk kell a biztonsági szintet. Az init(8) man oldalon olvashatunk részletesebben a biztonsági szintek mûködésérõl, illetve az /etc/defaults/rc.conf állományból és az rc.conf(5) man oldalról tudhatunk meg többet az rc.conf mûködésérõl.

10.24. Az rpc.statd parancsnak miért kell 256 MB memória?

Nem, itt szó sincs semmiféle memóriaszivárgásról, és egyébként sem használ 256 MB memóriát. Az `rpc.statd` parancs egyszerûen csak kényelmi megfontolásokból iszonyatos mennyiségû memóriát képez le a címterébe. Ebben technikailag semmi kivetnivaló nincsen, ezzel egyedül a top(1), ps(1) és a hozzá hasonló programokat zavarja meg egy kicsit.

A rpc.statd(8) tehát leképezi az állapotát rögzítõ állományt (amely a /var könyvtárban található a címterébe. Ilyenkor igyekszik egy kicsit elõre gondolkodni és felkészülni a megnövekedésére, ezért viszonylag nagy méretben hozza létre ezt a leképezést. Ezt nagyon jól megfigyelhetjük a forráskódjából is, ahol látszik, hogy a mmap(2) függvényt a `0x10000000` értékkel hívja meg, tehát az 32 bites Intel architektúrán megcímezhetõ memória egytizenhatod részével, ami pontosan 256 MB.

10.25. Miért nem törölhetõ az schg állományjelzõ?

Rendszerünkben a biztonsági szint (`securelevel`) nagyobb nullánál. Próbáljuk meg csökkenteni az értékét és próbálkozzunk ismét. Ezzel kapcsolatban részletesebb információkat a a biztonsági szintekrõl szóló kérdésbõl vagy az init(8) man oldalról tudhatunk meg.

10.26. Az .shosts állományon keresztül alapértelmezés szerint miért enged hitelesíteni a legújabb FreeBSD verziókban megtalálható SSH?

A legújabb FreeBSD verziókban azért nem tudjuk az .shosts állományon keresztül hitelesíteni magunkat, mert az ssh(1) alapértelmezés szerint rendszeradminisztrátori jogok nélkül kerül telepítésre. Ezt a "hibát" többféle módon ki tudjuk "javítani":

- Ha tartós megoldásra van szükségünk, akkor az /etc/make.conf állományban állítsuk az `ENABLE_SUID_SSH` változót a `true` értékre, majd fordítsuk újra az ssh(1) programot (vagy futtassuk le a `make world` parancsot).
- Ha ideiglenesen akarjuk csak javítani, akkor az /usr/bin/ssh állomány engedélyeit `root` felhasználóként állítsuk a `4555` értékre a `chmod 4555 /usr/bin/ssh` parancs kiadásával. Ezután vegyük fel az `ENABLE_SUID_SSH= true` sort az /etc/make.conf állományt, így ez a változtatás a `make world` következõ futtatásakor is megmarad.

10.27. Mi az a vnlru?

A `vnlru` törli és szabadítja fel a rendszerben keringõ vnode-okat, amikor a rendszermagban elérik a `kern.maxvnodes` változó által beállított határt. Ez a rendszermagban futó szál többnyire csak tétlenül ül a háttérben, és csak olyankor lép mûködésben, amikor rengeteg memóriát használunk és éppen több tízezernyi apró állományhoz akarunk egyszerre hozzáférni.

10.28. Mit jelentenek top parancs által megjelenített különbözõ memóriaállapotok?

- `Active` (Aktív): az utóbbi idõben használt lapok.
- `Inactive` (Inaktív): az utóbbi idõben nem használt lapok.
- `Cache` (Tárazott): (leginkább) azok a lapok, amelyeket még használnak, de gyakran azonnal újrafelhasználódnak (akár a régi, akár egy új hozzárendelésben). Egyes lapok az `active` állapotból közvetlenül a `cache` állapotba váltanak, ha tiszták (nem módosították), de ez az átmenet függ a házirendtõl, vagyis a VM alrendszer karbantartója által kiválasztott algoritmustól.
- `Free` (Szabad): effektív tartalom nélküli lapok, amelyek akár közvetlenül fel is használhatóak olyan esetekben, amikor a tárazott lapok erre nem alkalmasak. A szabad lapokat megszakításokban és a futó programokban is felhasználhatjuk.
- `Wired` (Rögzített): olyan lapok, amelyek a memória egy rögzített pontján foglalnak helyet. Ezeket többnyire a rendszermag használja, de speciális esetekben a programoknak is szükségük lehet rá.

A lapok általában akkor kerülnek ki a lemezre (valamilyen VM alrendszerbeli szinkronizáció során), amikor inaktív állapotban vannak, de akár az aktív lapok is szinkronizálhatóak. Ez attól függ, hogy a processzor képes-e nyomkövetni a lapok módosítását, és némely helyzetekben elõnyös lehet a rendszer számára, ha annak megfelelõen szinkronizálja a VM lapjait, hogy azok aktívak

vagy inaktívak. A legtöbb esetben itt egyszerûen csak egy olyan sort kell elképzelni, ahol a program számára viszonylag inaktív lapok találhatóak, amelyeket a rendszer tetszõlegesen a lemezre írhat. A tárazott lapok általában már eleve szinkronizáltak, nem leképzettek, közvetlenül a programok régi és új hozzárendelései használják ezeket. A szabad lapokat akár a megszakítások szintjén is lehet használni, miközben a tárazott vagy szabad lapokat a futó programokban érthetjük el. A tárazott lapok zárolása nem megfelelõ ahhoz, hogy megszakításokban is el lehessen érni ezeket.

Vannak még bizonyos jelzések (például a foglaltságot vagy foglaltság mértékét jelzõ értékek), amelyek még hatással vannak a fentebb leírt szabályokra.

10.29. Mekkora a rendelkezésre álló memória mérete?

A "rendelkezésre álló memóriának" rengeteg típusa létezik. Ezek közül egyik az a memória, amely közvetlenül anélkül elérhetõ, hogy bármi mást ki kellene hozzá lapoznunk. Ennek a mérete nagyjából a tárazott és a szabad lapokat tároló sorok hosszával arányos (amelyet még a rendszer beállításaitól függõ további tényezõk is módosíthatnak). A "rendelkezésre álló memória" másik típusa a teljes VM terület mérete. Ezt nem olyan könnyû meghatározni, de leginkább a lapozóterület és a fizikai memória méretétõl függ. A "rendelkezésre álló memória" több más lehetséges megfogalmazása is létezik, de szinte teljesen felesleges beszélni róluk. Egyedül az a fontos, hogy a igyekezzünk mérsékelni a lapozást és mindig legyen elegendõ lapozóterületünk.

10.30. Mi az a /var/empty? Nem lehet letörölni!

A /var/empty könyvtárat az sshd(8) program használja a privilégiumok elkülönítéséhez. A /var/empty könyvtárnak üresnek kell lennie, legyen a `root` tulajdonában és legyen rajta a `schg` állományjelzõ.

Noha semmiképpen sem javasoljuk a könyvtár törlését, úgy tudjuk elvégezni, ha elõször az `schg` állományjelzõt töröljük róla. A chflags(1) man oldalán olvashatunk ezzel kapcsolatban részletesebb információkat (azonban ne felejtsük el számításba venni az esetleges nehézségeket).

Chapter 11. Az X Window System és a virtuális konzolok használata

11.1. Mi az X Window System?

Az X Window System (vagy gyakran csak X11) a UNIX® és UNIX®-szerû operációs rendszereken, így többek közt a FreeBSD-n is az egyik leginkább elterjedt ablakozórendszer. A The X.Org Foundation felügyeli az X protokoll szabványait, azok aktuális referencia implementációival együtt. Ezek hivatalos megnevezése "Version 11 Release 7.7", de ezt gyakran csak X11 néven rövidítik.

Számos implementációja is elérhetõ több különbözõ architektúrára és operációs rendszerre. A protokoll szerver oldali funkcióit megvalósító programokat hivatalosan "X szervereknek" nevezik.

11.2. FreeBSD alatt milyen X implementációk használhatóak?

Kezdetben a FreeBSD alapértelmezett X implementációja az XFree86™ volt, amelyet a The XFree86 Project, Inc. tartott karban. Ez a változat volt használatban alapértelmezés szerint egészen a FreeBSD 4.10 és 5.2 verziójáig. Habár eközben az Xorg maga is karbantartotta a saját változatát, kizárólag csak referencia célokat használt és az évek során teljesen leromlott az állapota.

2004 elején azonban az XFree86 néhány korábbi fejlesztõje elhagyta a projektjüket, mivel nem értettek egyet bizonyos kérdésekben, például a forráskód ütemét, a jövõbeni irányokat és egyéb személyes konfliktusokat illetõen, és helyette közvetlenül az Xorg kódját kezdték el fejleszteni. Ekkor az Xorg hozzáigazította forrásait az utolsó XFree86™ kiadás forrásaihoz (XFree86 4.3.99.903), majd megváltoztatta a licencelését. és beolvasztott több, korábban külön karbantartott változtatást, aminek eredményeképpen végül megszületett az X11R6.7.0. Egy különálló, de velük együttmûködõ projekt, a freedesktop.org (vagy röviden csak `fd.o`) jelenleg is az eredeti XFree86™ források újraszervezésén dolgozik, aminek célja a napjainkban megjelenõ grafikus kártyák minél nagyobb mértékû kihasználása (és ezáltal a rendszer gyorsítása), a rendszer modularisabbá tétele (ezáltal a rendszer karbantarthatóságának javítása, ami a kiadások gyorsabb elõkészítését és könnyebb beállíthatóságát teszi lehetõvé). Az Xorg a jövõben tervezi a `freedesktop.org` fejlesztéseit is átvenni.

2004 júliusától kezdõdõen a FreeBSD-CURRENT változatban az XFree86™ helyett az Xorg lett az alapértelmezett X implementáció. A FreeBSD-ben azóta is alapból az Xorg X11 implementációja található meg.

A témával kapcsolatban a kézikönyv X11-rõl szóló fejezetében kaphatunk részletesebb felvilágosítást.

11.3. Mégis miért vált szét a két X projekt?

Ezt a kérdést ez a GYIK nem tudja megválaszolni. Ezzel kapcsolatban viszont érdemes elolvasnunk a különbözõ levelezési listák archívumait szerte az interneten. Keressünk rá a válaszra a kedvenc keresõnkben, de ezzel a kérdéssel ne a FreeBSD levelezési listáit zavarjuk. Az is elképzelhetõ, hogy

ennek a valós okait csak néhányan ismerik egész teljesen.

11.4. A FreeBSD miért az Xorg változatát választotta alapértelmezettnek?

Az Xorg fejlesztõi azt ígérték, hogy gyorsabban fognak újabb verziókat kiadni, amelyek sokkal több újítást is fognak tartalmazni. Nos, amennyiben tényleg állják a szavukat, azzal mindenki jól jár. Emellett az õ változatuk továbbra is a hagyományos X licenc alatt érhetõ el, miközben az XFree86™ licence ettõl némileg eltér.

11.5. Hogyan lehet használni az X-et?

Amennyiben már egy meglévõ rendszerre szeretnénk telepíteni az X-et, úgy érdemes a x11/xorg metaportot választanunk, amely magától feltelepíti az összes szükséges komponenst, vagy egyszerûen telepítsük az Xorg alkalmazást csomagból:

```
#  pkg_add -r xorg
```

Emellett az Xorg a sysinstall(8) használatával is telepíthetõ: válasszuk a Configure (Beállítások), Distributions (Terjesztések), végül a The X.Org Distribution (Az X.Org terjesztés) menüpontokat.

Az Xorg sikeres telepítése után kövessük a kézikönyv X11 beállításával foglalkozó szakaszában leírtakat.

11.6. Az X indításakor egy KDENABIO failed (Operation not permitted) hiba keletkezik, közvetlenül a startx parancs kiadása után. Mi lehet ezzel kezdeni?

A rendszerünkön valószínûleg túlságosan magas a biztonsági szint (`securelevel`) értéke. Ilyenkor az X-et nem tudjuk elindítani, mivel a mûködéséhez szüksége van a io(4) eszköz írására. Ezzel kapcsolatban az init(8) man oldal ad részletesebb útmutatást.

A kérdés tehát az, hogy mit kellene ezzel csinálni. Alapvetõen két lehetõségünk van: vagy visszaállítjuk a biztonsági szintet nullára (ezt általában az /etc/rc.conf állományon keresztül lehet megtenni), vagy az xdm(1) programot még a rendszerindítás során elindítjuk (mielõtt a biztonsági szintet magasabbra állítanánk).

A Hogyan indítható el az XDM a rendszer indításakor? szolgál arról bõvebb információval, hogy miként tudjuk használni az xdm(1) programot a rendszer indítása során.

11.7. Miért nem mûködik X alatt az egér?

Ha a syscons(4) (vagyis az alapértelmezett konzol) meghajtót használjuk, akkor be tudjuk úgy állítani a FreeBSD-t, hogy minden virtuális képernyõn látható legyen az egérkurzor. A syscons(4) egy /dev/sysmouse nevû virtuális eszköz támogatásával igyekszik elkerülni azt, hogy összeakadjon

az X-szel. A valós egértõl érkezõ összes eseményt a moused(8) démon írja folyamatosan a sysmouse(4) eszközre. Amennyiben az egerünket egy vagy több virtuális konzolon is használni akarjuk az X-szel *együtt*, akkor nézzük meg a Az egeret az X Window Systemen kívül is lehet valamilyen módon használni? válaszát és állítsuk be annak megfelelõen a moused(8) démont.

Ezt követõen nyissuk meg az /etc/X11/xorg.conf állományt és gondoskodjunk róla, hogy a következõ sorok feltétlenül szerepeljenek benne:

```
Section "InputDevice"
   Option          "Protocol" "SysMouse"
   Option          "Device" "/dev/sysmouse"
.....
```

Az Xorg 7.4 változatától kezdõdõen az xorg.conf állomány `InputDevice` szekciói nem kerülnek feldolgozásra a csatlakoztatott eszközök automatikus érzékelése esetén. A korábbi viselkedési mód visszaállításához vegyük fel a következõ sort a `ServerLayout` vagy `ServerFlags` szekciók valamelyikébe:

```
Option "AutoAddDevices" "false"
```

Néhányan inkább a /dev/mouse eszközt szeretik használni X alatt. Ha mi is így akarjuk használni, akkor a /dev/mouse eszközhöz hozzunk létre egy szimbolikus linket a /dev/sysmouse eszközre (lásd sysmouse(4)). Ezt úgy tudjuk megtenni, ha az /etc/devfs.conf állományba (lásd devfs.conf(5)) felvesszük a következõ sort:

```
link    sysmouse    mouse
```

A link maga közvetlenül a devfs(5) újraindításával keletkezik. Ehhez (`root` felhasználóként) a következõ parancsot kell kiadnunk:

```
# /etc/rc.d/devfs restart
```

11.8. X alatt lehet használni görgõs egeret?

Igen.

Jelezni kell az X-nek, hogy ötgombos egerünk van. Ezt úgy tudjuk megcsinálni, ha az /etc/X11/xorg.conf állományba felvesszük a `Buttons 5` és `ZAxisMapping 4 5` sorokat az "InputDevice" szakaszba. Vegyük például, hogy az /etc/X11/xorg.conf állományunkban a következõ "InputDevice" szakasz található.

Példa 1. Egy példa Xorg konfigurációs állomány "InputDevice" szakasza görgõs egerekhez

```
Section "InputDevice"
```

```
    Identifier      "Mouse1"
    Driver          "mouse"
    Option          "Protocol" "auto"
    Option          "Device" "/dev/sysmouse"
    Option          "Buttons" "5"
    Option          "ZAxisMapping" "4 5"
EndSection
```

Példa 2. Egy egyszerû példa ".emacs" állomány görgõs egerek (opcionális) használatához

```
;; görgõs egér
(global-set-key [mouse-4] 'scroll-down)
(global-set-key [mouse-5] 'scroll-up)
```

11.9. Hogyan lehet távoli X szervereket elérni?

Biztonsági okokból a szerver alapértelmezés szerint nem engedélyezi, hogy egy távoli géprõl ablakot lehessen nyitni rajta.

Ha szükségünk lenne erre a lehetõségre, akkor nem kell mást tennünk, mint az X-et a `-listen_tcp` paraméterrel indítani:

```
% startx -listen_tcp
```

11.10. Mi az a virtuális konzol és hogyan lehet belõle többet létrehozni?

A virtuális konzolok röviden szólva arra alkalmasak, hogy egyetlen gépen is több párhuzamos munkamenetben tudjunk dolgozni, hálózat vagy X beállítása nélkül.

Amikor a rendszer elindul, a rendszerüzenetek után általában egy bejelentkezõ képernyõ jelenik meg. Ekkor az elsõ virtuális konzolon keresztül tudjuk megadni a felhasználói nevünket és jelszavunkat, majd nekilátni a munkának (vagy éppen a játszadozásnak).

Késõbb aztán elõfordulhat, hogy egy másik munkamenetet is szeretnénk elindítani, például elõkeresni az éppen használt program dokumentációját vagy elolvasni a leveleinket, amíg FTP-n keresztül letöltünk egy állományt. Ehhez nem kell mást csinálnunk, csak le kell nyomni az `Alt` + `F2` (tartsuk lenyomva az `Alt` billentyût miközben megnyomjuk az `F2` billentyût) billentyûkombinációt és máris egy másik virtuális konzolon találjuk magunkat! Ha innen vissza szeretnénk térni az elõzõ munkamenetbe, akkor nyomjuk le az `Alt` + `F1` billentyûkombinációt.

A frissen telepített FreeBSD rendszerekben alapértelmezés szerint nyolc virtuális konzol engedélyezett. Az `Alt` + `F1`, `Alt` + `F2`, `Alt` + `F3`, stb. lenyomásával tudunk váltogatni köztük.

Ha ennél többet szeretnénk egyszerre használni, akkor nyissuk meg az /etc/ttys állományt (lásd ttys(5)) és a "Virtual terminals" részben vegyünk még fel a ttyv8 eszköz után továbbiakat, egészen a ttyvc eszközig:

```
# Írjuk át az eredeti ttyv8 bejegyzést az /etc/ttys
# állományban és engedélyezzük.
ttyv8   "/usr/libexec/getty Pc"         cons25  on secure
ttyv9   "/usr/libexec/getty Pc"         cons25  on secure
ttyva   "/usr/libexec/getty Pc"         cons25  on secure
ttyvb   "/usr/libexec/getty Pc"         cons25  on secure
```

Akármennyit használhatunk belõlük. Ne felejtsük el azonban, hogy minél több virtuális terminálunk van, annál több erõforrásra lesz hozzájuk szükségünk. Ezt leginkább akkor érdemes megfontolni, ha 8 MB memóriánál kevesebbel rendelkezünk. Emellett még érdemes a `secure` értéket is az `insecure` értékre átállítani.

Ha X szervert is akarunk futtatni, akkor legalább egy virtuális konzolt szabadon (vagy kikapcsolva) *kell* hagynunk a számára. Így tehát, ha mind a tizenkét funkcióbillentyûre szeretnénk elindítani egy-egy virtuális konzolt, nos, akkor nincs szerencsénk - ha X szervert is akarunk használni a gépen, akkor legfeljebb csak tizenegyet használhatunk belõlük.

Az egyes konzolokat legegyszerûbben úgy tudjuk letiltani, ha kikapcsoljuk ezeket. Például, ha az elõbb említettek szerint tizenkét terminálunk van, és X-et akarunk futtatni, akkor a tizenkettedik terminál beállításait meg kell változtatnunk errõl:

```
ttyvb   "/usr/libexec/getty Pc"         cons25  on  secure
```

erre:

```
ttyvb   "/usr/libexec/getty Pc"         cons25  off secure
```

Amennyiben a billentyûzetünkön csak tíz funkcióbillentyû található, elengedõ ennyi is:

```
ttyv9   "/usr/libexec/getty Pc"         cons25  off secure
ttyva   "/usr/libexec/getty Pc"         cons25  off secure
ttyvb   "/usr/libexec/getty Pc"         cons25  off secure
```

(Ezeket a sorokat akár ki is törölhetjük.)

Ezt követõen a legegyszerûbben (és egyben a legtisztábban) úgy tudjuk aktiválni a virtuális konzolokat, ha újraindítjuk a rendszerünket. Ha viszont nem akarjuk ezt feltétlenül megtenni, akkor állítsuk le az X szervert, majd (`root` felhasználóként) adjuk ki az alábbi parancsot:

```
# kill -HUP 1
```

Fontos, hogy a parancs végrehajtás elõtt teljesen leállítsuk az X szervert, amennyiben az fut. Ha nem tesszük meg, akkor könnyen elõfordulhat, hogy a `kill` parancs hatására lemerevedik vagy megáll a rendszerünk.

11.11. Hogyan lehet elérni a virtuális konzolokat X-bõl?

A virtuális konzolokra a `Ctrl` + `Alt` + `FN` billentyûkombinációval lehet visszaváltani. Ennek megfelelõen tehát a `Ctrl` + `Alt` + `F1` kombinációval az elsõ virtuális konzolra tudunk visszaváltani.

Ahogy visszajutottunk a szöveges konzolra, az `Alt` + `Fn` billentyûkombinációval a megszokott módon tudunk váltani köztük.

Ha innen az X szerverre akarunk visszaváltani, akkor egyszerûen csak váltsunk arra a virtuális konzolra, ahol az X fut. Ha az X-et a paranccsorból indítottuk el (például a `startx` paranccsal), akkor az X nem arra a virtuális konzolra kapcsolódik automatikusan, amelyen a parancsot kiadtuk, hanem az utána következõ, használatban még nem levõ konzolra. Ha nyolc aktív virtuális terminálunk van, akkor az X a kilencediken fog futni, ezért ide az `Alt` + `F9` lenyomásával tudunk visszatérni.

11.12. Hogyan indítható el az XDM a rendszer indításakor?

Alapvetõen kétféle megközelítés létezik az xdm(1) elindításával kapcsolatban. Az egyik megközelítés szerint az `xdm` parancsot az /etc/ttys állományból (lásd ttys(5)) tudjuk megadni a megadott példa alapján, a másikban pedig egyszerûen az rc.local állományból (lásd rc(8)) vagy a /usr/local/etc/rc.d könyvtárban megadható X szkripttel. Mind a kettõ ugyanazt képviseli, de vannak bizonyos helyzetek, ahol a kettõ közül csak az egyik mûködik. Az eredmény mind a két esetben azonos, hatásukra az X egy grafikus bejelentkezõ képernyõvel jelentkezik.

A ttys(5) módszernek van egy olyan elõnye, hogy pontosan megadja, melyik virtuális terminálon fog futni az X és a szerver elindítását az init(8) programra bízza. Az rc(8) használata esetén viszont könnyû leállítani az `xdm` programot, ha netalán valamilyen gondunk adódna az X szerver indításakor.

Ha az rc(8) állományból töltöttük be, akkor az `xdm` futtatásához semmilyen paramétert nem kell megadni (például, hogy démonként fusson). Az xdm(1) azonban csak az *összes* getty(8) elindulása után indítható, máskülönben a két program ütközni fog és a konzol nem tud létrejönni. Ezt a legkönnyebben úgy lehet megakadályozni, ha az `xdm` indítása elõtt várunk kb. 10 másodpercet a szkriptben.

Amennyiben az /etc/ttys állományból adjuk ki az `xdm` parancsot, úgy továbbra is fennáll az xdm(1) és a getty(8) ütközésének veszélye. Ezt például úgy tudjuk elkerülni, ha felvesszük a megfelelõ virtuális terminál sorszámát a /usr/local/lib/X11/xdm/Xservers állományba:

```
:0 local /usr/local/bin/X vt4
```

A fenti példában az X szervert a /dev/ttyv3 eszközre irányitjuk. A számozást azonban eggyel el kell tolnunk, mert míg az X szerver egytõl számozza a virtuális konzolokat, addig a FreeBSD rendszermagja nullától.

11.13. Az xconsole indításakor miért jelenik meg a Couldn't open console hibaüzenet?

Ha az X-et a `startx` paranccsal indítottuk el, akkor a /dev/console eszközre *nem* állítódnak be a szükséges engedélyek, ezért az `xterm -C` és az `xconsole` parancsok nem fognak mûködni.

Ez a konzolok engedélyeinek alapértelmezett beállítási módjától függ. Egy többfelhasználós rendszer esetén nem feltétlenül van szükségünk arra, hogy bármelyik felhasználó kedvére írhasson a rendszerkonzolra. Az fbtab(5) állomány segítségével engedélyezni tudjuk azon felhasználók számára, akik a helyi gépen, virtuális konzolon keresztül jelentkeznek be.

Dióhéjban az /etc/fbtab állományban (lásd fbtab(5)) kell kivennünk a következõ sort a megjegyzésbõl:

```
/dev/ttyv0 0600 /dev/console
```

Ennek köszönhetõen bárki, aki az /dev/ttyv0 eszközön keresztül jelentkezik be a rendszerbe, el tudja érni a konzolt.

11.14. Régebben egyszerû felhasználóként is el lehetett indítani az XFree86™ szervert. Most miért kell root felhasználóként indítani?

Az X szerverek csak úgy képesek közvetlenül elérni a videokártyát, ha `root` felhasználóként futtatjuk ezeket. Az XFree86™ régebbi (3.3.6 elõtti) változatai az összes szervert úgy telepítették fel automatikusan, hogy a `root` felhasználó jogaival fussanak (setuid bittel). Ennek viszont megvan a maga nyilvánvaló biztonsági kockázata, hiszen az X szerverek általában nagy és bonyolult programok. Az XFree86™ újabb változatai azonban már pontosan ebbõl kifolyólag nem állítanak be setuid `root` bitet a szerverekre.

Értelemszerûen az a megoldás nem fogadható el és nem is annyira biztonságos, hogy az X szervert `root` felhasználóként futtassuk. Kétféleképpen tudjuk egyszerû felhasználóként futtatni az X-et. Használhatjuk az `xdm` vagy más egyéb bejelentkeztetõ képernyõ (mint például a `kdm`) megoldását, vagy az `Xwrapper` programot.

Az `xdm` egy grafikus bejelentkeztetésért felelõs démon. Általában a rendszer indításakor aktiválódik, feladata a felhasználók hitelesítése és a hozzájuk tartozó munkamenetek elindítása. Lényegében a getty(8) és a login(1) grafikus megfelelõje. Az `xdm` démonnal kapcsolatban még az XFree86™

dokumentációját, illetve a GYIK-ban ezt a kérdést érdemes elolvasnunk.

Az `Xwrapper` az X szerverhez tartozó burkolóprogram (wrapper). Ez egy apró segédprogram, amely lehetõvé teszi az X szerver manuális indítását miközben igyekszik ügyelni a biztonságra is. Elvégez néhány alapvetõ ellenõrzést a paramétereken, és ha megfelelõnek találja ezeket, akkor elindítja a megfelelõ X szervert. Ha valamiért nem akarunk bejelentkeztetõ képernyõt indítani, akkor ezt pontosan nekünk találták ki! Ha telepítettük a teljes Portgyûjteményt, akkor a /usr/ports/x11/wrapper portban találjuk meg.

11.15. Miért viselkednek furcsán a PS/2-es egerek X alatt?

Valószínûleg az egér és az egérmeghajtó kiesett a szinkronból.

Nagyon ritkán elõfordul, hogy a meghajtó hibásan szinkronizációs hibát jelez, és ekkor a rendszermag a következõ üzenetet küldi:

```
psmintr: out of sync (xxxx != yyyy)
```

Közben természetesen azt tapasztaljuk, hogy az egerünk nem mûködik rendesen.

Ha ilyen történne velünk, akkor tiltsuk le a meghajtó szinkronizáció ellenõrzéséért felelõs rutinjait. Ezt úgy tudjuk megtenni, ha a meghajtónak beállítjuk a `0x100` értéket. Ehhez a rendszertöltõ parancssorában a `-c` kapcsolóval tudjuk behozni a *UserConfig* részt:

```
boot: -c
```

Ezután a *UserConfig* parancssorában gépeljük be a következõt:

```
UserConfig> flags psm0 0x100
UserConfig> quit
```

11.16. Miért nem mûködnek a MouseSystems által gyártott PS/2-es egerek?

Kaptunk néhány visszajelzést arra vonatkozóan, hogy a MouseSystems által gyártott PS/2-es egerek bizonyos típusai csak abban az esetben mûködnek rendesen, ha "nagy felbontású" módban használjuk ezeket. Minden más esetben az egér néha fel-felugrik a képernyõ bal felsõ sarkába.

Úgy tudjuk nagy felbontású módban használni az egerünket, ha a PS/2-es egérmeghajtónak a `0x04` beállítást adjuk meg. Ehhez a rendszertöltõ parancssorában gépeljük be a `-c` kapcsolót:

```
boot: -c
```

Ahogy bejön a *UserConfig* parancssora, gépeljük be a következõt:

```
UserConfig> flags psm0 0x04
UserConfig> quit
```

Az elõzõ részben olvashatunk egy másik hasonló egeres problémáról.

11.17. Hogyan lehet megcserélni a gombokat az egéren?

Futtassuk le a `xmodmap -e "pointer = 3 2 1"` parancsot az .xinitrc vagy .xsession állományunkból.

11.18. Hogyan lehet betöltõképet telepíteni és hol találhatóak ilyen képek?

Erre a kérdésre részletes választ a FreeBSD kézikönyv Rendszerbetöltõ képernyõk címû szakaszában kapunk.

11.19. X alatt lehet használni a billentyûzeten található Windows billentyûket?

Igen. Ehhez mindössze az xmodmap(1) használatával meg kell adni a hozzájuk tartozó funkciót.

Feltéve, hogy mindegyik "Windows" billentyûzet szabványos, a következõ billentyûkódok tartoznak ehhez a három plusz gombhoz:

- 115 - `Windows` billentyû, a bal oldali `Ctrl` és `Alt` billentyûk között
- 116 - `Windows` billentyû, az `AltGr` mellett jobbra
- 117 - `Menü` gomb, a jobb oldali `Ctrl` mellett balra

Például így lehet beállítani a bal oldali `Windows` billentyût vesszõre:

```
# xmodmap -e "keycode 115 = comma"
```

A változatatások valószínûleg csak akkor fognak életbelépni, ha újraindítjuk az ablakkezelõnket.

Ha azt szeretnénk, hogy a `Windows` billentyûkhöz rendelt funkciók az X indításakor automatikusan beállítódjanak, akkor tegyük az `xmodmap` parancs hívását az ~/.xinitrc állományunkba. Sokkal jobban járunk viszont, ha ehelyett inkább az ~/.xmodmaprc állományunkba vesszük fel az `xmodmap` beállításait, soronként egyesével, és a következõ sor tesszük az ~/.xinitrc állományunkba:

```
xmodmap $HOME/.xmodmaprc
```

Például ezeket a gombokat be lehet állítani az `F13`, `F14` és `F15` billentyûkre is. Ezekre aztán az alkalmazásokban vagy az ablakkezelõben további hasznos funkciókat tudunk beállítani.

Ehhez a következõt kell megadnunk az ~/.xmodmaprc állományban:

```
keycode 115 = F13
keycode 116 = F14
keycode 117 = F15
```

Ha például az x11-wm/fvwm2 ablakkezelõt használjuk, akkor az `F13` gombra be tudjuk állítani a kurzor alatt álló ablak lekicsinyítésére (vagy visszanagyítására); az `F14` billentyûvel az elõtérbe tudjuk hozni a kurzor alatt levõ ablakot, vagy ha már elöl van, akkor hátra tudjuk rakni; az `F15` gomb elõhozza a munkakörnyezet (alkalmazás) menüjét még olyankor is, amikor a kurzor nincs is az asztalon. Ez utóbbi abban az esetben lehet hasznos, amikor az asztal egyáltalán nem látható (és a billentyûn látható rajz pontosan is ezt mutatja).

A következõ beállítások valósítják meg az imént említett funkciókat az ~/.fvwmrc állományon belül:

```
Key F13         FTIWS    A        Iconify
Key F14         FTIWS    A        RaiseLower
Key F15         A        A        Menu Workplace Nop
```

11.20. Hogyan lehet hardveres 3D gyorsítást használni az OpenGL®-hez?

Az Xorg pillanatnyilag használt verziójától és a videokártyánktól függ, hogy tudunk-e 3D gyorsítást alkalmazni. Ha nVidia kártyánk van, akkor a portok közül telepíteni tudjuk a FreeBSD-hez készített bináris meghajtót:

- A legújabb nVidia-kártyákat az x11/nvidia-driver port támogatja.
- A GeForce2 MX/3/4 sorozatú nVidia-kártyákat a meghajtó 96XX változata támogatja, amely az x11/nvidia-driver-96xx portból telepíthetõ.
- Az ettõl is régebbi kártyák, például a GeForce vagy Riva TNT esetén a meghajtó 71XX változata javasolt, amely az x11/nvidia-driver-71xx porton keresztül érhetõ el.

Az nVidia honlapján részletes leírást találhatunk arról, hogy melyik kártyát melyik meghajtó ismeri. Ez az információ a következõ címen érhetõ el: http://www.nvidia.com/object/IO_32667.htm.

A Matrox G200/G400 esetén az x11-servers/mga_hal portot érdemes megnéznünk.

ATI Rage 128 és Radeon kártyák számára a ati(4), r128(4) és radeon(4) man oldalakat ajánljuk.

3dfx Voodoo 3, 4, 5 és Banshee kártyák számára az x11-servers/driglide port áll rendelkezésre.

Chapter 12. Hálózatok

12.1. Honnan lehet többet megtudni a lemez nélküli mûködésrõl?

A "lemez nélküli mûködés" kifejezés arra utal, hogy a FreeBSD rendszerünk hálózaton keresztül indul el, valamint a mûködéséhez szükséges állományokat nem merevlemezrõl, hanem egy szerverrõl olvassa be. Ennek részleteirõl kézikönyv lemez nélküli mûködésrõl szóló részében olvashatunk.

12.2. A FreeBSD használható kizárólag csak hálózati útválasztóként?

Igen. Ezzel kapcsolatban a kézikönyv Egyéb haladó hálózati témák címû fejezetét javasoljuk elolvasásra, különös tekintettel az útválasztás és az átjárók bemutatására.

12.3. FreeBSD-n keresztül lehet Windows® operációs rendszerrel internetre csatlakozni?

Ezt a kérdést általában olyanok teszik fel, akiknek két számítógépük van otthon, és ezek közül az egyiken a FreeBSD, a másikon pedig a Windows® valamelyik változata fut. A FreeBSD rendszer fog az internethez csatlakozni, és ezen keresztül szeretnénk a windowsos géprõl is elérni azt. Ez tulajdonképpen az elõzõ kérdés egy speciális esete, és remekül megoldható.

Ha betárcsázós kapcsolattal csatlakozunk az internethez, akkor érdemes tudnunk, hogy a felhasználói módban futó ppp(8) tartalmaz egy `-nat` kapcsolót. A ppp(8) programot úgy tudjuk a `-nat` kapcsolóval futtatni, ha az /etc/rc.conf állományban a `gateway_enable` beállítást a *YES* értékre állítjuk. Ezután állítsuk be a windowsos gépünket ennek megfelelõen és minden mûködni fog. A további részletekrõl a ppp(8) man oldalán vagy a kézikönyv felhasználói PPP-rõl szóló bejegyzésében olvashatunk.

Amennyiben rendszerszintû PPP-t használunk vagy Ethernettel csatlakozunk az internethez, akkor a natd(8) démonra lesz szükségünk. Erre vonatkozóan a kézikönyv natd démonról szóló szakaszában találhatunk részletesebb információkat.

12.4. A FreeBSD támogatja a SLIP és a PPP használatát?

Igen. Érdemes elolvasnunk az slattach(8), sliplogin(8), ppp(8) és pppd(8) man oldalakat. A ppp(8) és a pppd(8) egyaránt támogatja a beérkezõ és kimenõ kapcsolatokat, miközben a sliplogin(8) kizárólag csak beérkezõ kapcsolatokkal dolgozik, valamint a slattach(8) pedig csak kimenõ kapcsolatokkal.

Ezek pontos használatáról olvassuk el a kézikönyv PPP-rõl és a SLIP-rõl szóló fejezetét.

Ha viszont csak egy "shellen" keresztül érjük el az internetet, akkor hasznos lehet megnéznünk a

net/slirp csomagot. Segítségével a helyi géprõl (korlátozott módon) hozzá tudunk férni különbözõ FTP és HTTP szolgáltatásokhoz.

12.5. A FreeBSD támogat hálózati címfordítást (NAT) vagy maszkolást?

Igen. Ha egy felhasználói PPP kapcsolat esetén szeretnénk hálózati címfordítást alkalmazni, akkor olvassuk el a kézikönyv felhasználói PPP-vel foglalkozó részét. Ha viszont más típusú hálózati kapcsolatok esetén kívánunk címfordítást használni, akkor ahhoz a kézikönyv natd démonnal kapcsolatos részét kell fellapoznunk.

12.6. A PLIP segítségével hogyan tudok két FreeBSD rendszert összekapcsolni párhuzamos porton keresztül?

Ezt illetõen a kézikönyv PLIP-rõl szóló szakaszát érdemes megnéznünk.

12.7. Hogyan lehet álneveket megadni az Ethernet eszközöknek?

Amennyiben az álnév ugyanazon az alhálózaton található, mint a hozzá tartozó interfész, akkor egyszerûen csak adjuk meg a `netmask 0xffffffff` paramétert az ifconfig(8) parancs meghívásakor, például így:

```
# ifconfig ed0 alias 192.0.2.2 netmask 0xffffffff
```

Minden más esetben a hagyományos módon adjunk meg egy hálózati címet és egy hálózati maszkot:

```
# ifconfig ed0 alias 172.16.141.5 netmask 0xffffff00
```

Errõl bõvebben a FreeBSD kézikönyvben olvashatunk.

12.8. A 3C503 kártya hogyan állítható másik hálózati portra?

Ha a kártyán egy másik portot szeretnénk használni, akkor ahhoz meg kell adnunk egy további paramétert a ifconfig(8) meghívásakor. Itt az alapértelmezett port a `link0`. Ha a BNC helyett az AUI portot akarjuk használni, akkor ennek a `link2` értéket kell megadnunk. Az ilyen típusú beállítások az /etc/rc.conf állomány (lásd rc.conf(5)) `ifconfig_*` változóival adhatóak meg.

12.9. Miért okoz gondot az NFS használata FreeBSD alatt?

A személyi számítógépekben található bizonyos hálózati kártyák (szépen szólva) ügyesebbek a többieknél, ami az olyan komolyabb hálózati alkalmazások, mint például az NFS esetén gondokat okozhat.

Ezzel kapcsolatban kézikönyv NFS-rõl szóló részét érdemes megnéznünk.

12.10. Miért nem lehet hálózati állományrendszereket csatlakoztatni Linux® alól?

A Linux® egyes változataiban található NFS kód csak bizonyos privilegizált portokról fogad el kéréseket. Próbáljuk meg a következõt:

```
# mount -o -P linux:/valami /mnt
```

12.11. Miért nem lehet hálózati állományrendszereket csatlakoztatni Sun™ típusú rendszerek alól?

A SunOS™ 4.*X* változatait futtató munkaállomások csak privilegizált portokról fognak el kéréseket. Próbálkozzunk az alábbi paranccsal:

```
# mount -o -P sun:/valami /mnt
```

12.12. A mountd miért küld folyton can’t change attributes hibaüzenetet és miért jelenik meg a bad exports list hibaüzenet a FreeBSD alapú NFS szerveren?

Ez leginkább azért történik, mert nem jól adtuk meg az /etc/exports állomány tartalmát. Olvassuk át a exports(5) man oldalt és a kézikönyv NFS-rõl szóló részét, különös tekintettel az NFS beállítására.

12.13. A NeXTStep gépekkel miért nem sikerül PPP-n keresztül kommunikálni?

Próbáljuk meg az /etc/rc.conf állományban (lásd rc.conf(5)) kikapcsolni a TCP kiterjesztések használatát úgy, hogy az alábbi változót a `NO` értékre állítjuk:

```
tcp_extensions=NO
```

A *Xylogic* által gyártott *Annex* típusú gépek esetén is javasolt megtenni a fenti változtatást.

12.14. Hogyan lehet engedélyezni a multicast használatát az IP-n belül?

A FreeBSD alapértelmezés szerint támogatja a multicast mûveleteket. Amennyiben egy multicast küldéseket közvetítõ útválasztót szeretnénk beállítani, akkor újra kell fordítanunk a rendszermagunkat a `MROUTING` beállítás használatával és elindítani a mrouted(8) démont. Ez a démon úgy aktiválható a rendszer minden egyes indításakor, ha az /etc/rc.conf állományban az `mrouted_enable` változót `YES` értékûre állítjuk.

A FreeBSD újabb változataiban az mrouted(8) multicast útválasztó démon, a mapmbone(8) valamint az mrinfo(8) segédprogramok már nem szerepelnek az alaprendszerben. Ezek a programok már a FreeBSD Portgyûjteményében az net/mrouted portban találhatóak meg.

Az MBONE használatához további eszközök találhatóak a külön mbone kategóriában a Portgyûjteményen belül. Ha a `vic` és `vat` nevû konferenciaszervezõ eszközöket keressük, akkor itt érdemes szétnéznünk!

12.15. Milyen hálózati kártyák épülnek a DEC PCI chipkészletére?

Glen Foster (gfoster@driver.nsta.org) a következõ listát állította össze róluk, amelyet kiegészítettünk még néhány további elemmel:

Táblázat 2. A DEC PCI chipkészletére épülõ hálózati kártyák

Gyártó	Típus
ASUS	PCI-L101-TB
Accton	ENI1203
Cogent	EM960PCI
Compex	ENET32-PCI
D-Link	DE-530
Dayna	DP1203, DP2100
DEC	DE435, DE450
Danpex	EN-9400P3
JCIS	Condor JC1260
Linksys	EtherPCI

Gyártó	Típus
Mylex	LNP101
SMC	EtherPower 10/100 (Model 9332)
SMC	EtherPower (Model 8432)
TopWare	TE-3500P
Znyx (2.2.*x*)	ZX312, ZX314, ZX342, ZX345, ZX346, ZX348
Znyx (3.*x*)	ZX345Q, ZX346Q, ZX348Q, ZX412Q, ZX414, ZX442, ZX444, ZX474, ZX478, ZX212, ZX214 (10mbps/hd)

12.16. Miért kell teljes hálózati neveket megadni?

Erre a FreeBSD kézikönyvben találjuk meg a választ.

12.17. Miért jelenik meg a Permission denied hibaüzenet minden egyes hálózati mûvelet esetén?

Amennyiben a rendszermagot az `IPFIREWALL` beállítással fordítottuk le, akkor nem szabad elfelejtenünk, hogy ez alapértelmezés szerint minden olyan csomagot eldob, amelyet külön nem engedélyeztünk.

Ha véletlenül rosszul állítottuk volna be a rendszerünkön futó tûzfalat, akkor a hálózat mûködését úgy tudjuk visszaállítani, ha `root` felhasználóként kiadjuk a következõ parancsot:

```
# ipfw add 65534 allow all from any to any
```

Az /etc/rc.conf állományban is megadhatjuk ehhez a `firewall_type="open"` sort.

Ha a tûzfalak beállításáról szeretnénk többet megtudni FreeBSD alatt, akkor olvassuk el a kézikönyv erre vonatkozó fejezetét.

12.18. Az ipfw fwd szabálya miért nem irányít át más gépekre szolgáltatásokat?

Valószínûleg azért, mert nem egyszerûen a csomagok továbbítására (forward) van szükségünk, hanem hálózati címfordításra. Az "fwd" szabály pontosan azt csinálja, amirõl a nevét kapta: csomagokat továbbít, de azokon belül semmit sem változtat meg. Tegyük fel, hogy van egy ilyen szabályunk:

```
01000 fwd 10.0.0.1 from any to ize 21
```

Amikor egy csomag az *ize* célcímmel megérkezik a *10.0.0.1* gépre, akkor benne a cél címe továbbra

is az *ize* lesz! A csomag célcíme *nem* fog magától megváltozni a *10.0.0.1* címre. A legtöbb gép általában eldobja azokat a csomagokat, amelyeket nem egyenesen neki címeztek. Emiatt a "fwd" szabály használata nem minden esetben úgy mûködik, ahogy arra a felhasználó számít. Ez viszont ilyen, semmilyen hiba nincs benne.

Részletesebb információkat a szolgáltatások átirányításával foglalkozó GYIK-ban, a natd(8) man oldalán vagy a Portgyûjtemény valamelyik port átirányítással foglalkozó portjának dokumentációjában találhatunk.

12.19. Hogyan lehet egyik géprõl a másikra szolgáltatásokat átirányítani?

Az FTP (vagy más egyéb szolgáltatás-) kéréseket a Portgyûjteményen belül található sysutils/socket porttal tudunk átirányítani. Az adott szolgáltatás helyett egyszerûen csak adjuk meg a `socket` parancsot és annak paramétereit, valahogy így:

```
ftp stream tcp nowait nobody /usr/local/bin/socket socket ftp.minta.com ftp
```

ahol az *ftp.minta.com* az a gép, ahová át akarjuk irányítani a szolgáltatást, az *ftp* pedig a konkrét szolgáltatás.

12.20. Hogyan lehet a sávszélességet szabályozni?

FreeBSD alatt alapvetõen három eszköz szolgál erre a célra. A dummynet(4) a FreeBSD részeként megtalálható ipfw(4) egyik komponense. Az ALTQ a FreeBSD-ben található pf(4) rendszer része, az Emerging Technologies által fejlesztett Bandwith Manager pedig egy kereskedelmi termék.

12.21. Miért jelenik meg a /dev/bpf0: device not configured hibaüzenet?

Olyan programot akarunk futtatni, amelynek szüksége van a Berkeley Packet Filter (bpf(4)) használatára, azonban a rendszermag ezt nem tartalmazza. Úgy tudjuk aktiválni, ha a rendszermag konfigurációs állományába felvesszük a következõ sort, majd fordítunk egy új rendszermagot:

```
device bpf        # Berkeley Packet Filter
```

12.22. Hogyan lehet a hálózaton elérhetõ Windows® típusú partíciókat csatlakoztatni, mint ahogy az smbmount csinálja Linux® alatt?

Erre az SMBFS eszközeit használhatjuk, amely tartalmazza az ehhez szükséges rendszermagbeli módosításokat és a hozzá tartozó felhasználói programokat. Ezek a programok és a hozzájuk

tartozó mount_smbfs(8) man oldal az alaprendszer részei.

12.23. Mik azok az Limiting icmp/open port/closed port response üzenetek a naplókban?

Ilyen üzeneteket akkor kapunk a rendszermagtól, ha valaminek a hatására több ICMP vagy TCP reset (RST) választ küld, mint amennyit kellene. Az ICMP válaszok sokszor olyankor generálódnak, amikor használaton kívüli UDP portokat akarnak elérni a rendszerünkön. A TCP reset pedig általában olyankor keletkezik, amikor meg nem nyitott TCP porthoz akarnak csatlakozni. Többek közt ilyenek okozhatják:

- A rendszer túlterhelését célzó, nyers erõvel indított *Denial of Service* (Dos) támadások (ellentétben az egycsomagos, adott sebezhetõség kihasználó támadásokkal).
- A portok szisztematikus letapogatása, amelynek során egyszerre nagy mennyiségû portot próbálnak meg átvizsgálni (ellentétben azzal, amikor csak néhány jól ismert portot nyitnak meg).

Az üzenetben olvasható elsõ szám azt mondja meg, hogy a rendszermag mennyi csomagot küldött volna, ha nem korlátoztuk volna, a második pedig magát a határt jelzi. Ezt a `net.inet.icmp.icmplim` sysctl változó segítségével tudjuk beállítani, ahogy például most megnöveljük az értékét `300`-ra:

```
# sysctl -w net.inet.icmp.icmplim=300
```

Amennyiben le szeretnénk tiltani az ilyen jellegû üzeneteket a naplókban, viszont még továbbra is szükségünk lenne a válaszküldés korlátozására, a `net.inet.icmp.icmplim_output` sysctl változó segítségével így tudjuk ezt megtenni:

```
# sysctl -w net.inet.icmp.icmplim_output=0
```

Végezetül, ha teljesen ki akarjuk kapcsolni a válaszküldés korlátozását, akkor állítsuk a `net.inet.icmp.icmplim` sysctl változót (lásd az elõbbi példában) a `0` nulla értékre. A korlát törlése azonban a fenti okok miatt egyáltalán nem ajánlott.

12.24. Mik azok az arp: unknown hardware address format hibaüzenetek?

Ez arra utal, hogy valamelyik gép a helyi Ethernet-alapú hálózatunkon olyan MAC-címet használ, amelynek a FreeBSD nem ismeri a formátumát. Valószínûleg olyankor kapjuk ezt a hibaüzenetet, amikor valaki más kísérletezik az Ethernet kártyája beállításaival valahol a hálózaton. Leggyakrabban kábelmodemes hálózatokon tapasztalhatunk ilyet. Megnyugodhatunk, teljesen veszélytelen, semmilyen hatással nincs a FreeBSD gépünk teljesítményére.

12.25. Miért jelennek meg 192.168.0.10 is on fxp1 but got reply from 00:15:17:67:cf:82 on rl0 üzenetek a konzolon és hogyan lehet ezeket kikapcsolni?

Ilyen üzeneteket akkor kapunk, amikor a hálózaton kívülrõl érkezik hozzánk váratlanul egy csomag. A letiltásukhoz állítsuk a `net.link.ether.inet.log_arp_wrong_iface` értékét `0`-ra.

12.26. A CVSup programot telepítése után nem lehet elindítani, mert hibákat jelez. Mi a gond?

Elõször is nézzük meg, hogy az iménti hibaüzenet mellett nem látunk-e valami hasonlót:

```
/usr/libexec/ld-elf.so.1: Shared object "libXaw.so.6" not found
```

Az ilyen jellegû hibák általában olyankor keletkeznek, amikor olyan gépre telepítjük a net/cvsup portot, amelyen viszont nem található meg a Xorg programcsomag. Amennyiben szükségünk lenne CVSup programhoz mellékelt grafikus felületre, akkor kénytelenek leszünk mellé az Xorg programjait is telepíteni. Ha viszont egyszerûen csak parancssorból szeretnénk használni a CVSup lehetõségeit, töröljük le a korábban telepített csomagot, majd helyette rakjuk fel a net/cvsup-without-gui vagy a net/csup portot. A FreeBSD újabb változataiban megpróbálkozhatunk a csup(1) használatával is. Ezzel a témával részletesebben a kézikönyv CVSup használatáról szóló része foglalkozik.

Chapter 13. Biztonság

13.1. Mi az a járóka (sandbox)?

A járóka alapvetõen egy biztonsági szakkifejezés. Két dolgot jelenthet:

- Egy virtuális falak között futó programot, melyeket azért emeltek a program köré, hogy a feltörését követõen megakadályozzák a rendszer többi részének elérését.

  A program csak a falon belül "játszhat". Ilyenkor semmilyen olyan kódot nem képes futtatni, amellyel át tudna lépni a falakon, így a használatához nem kell elõzetesen átvizsgálni a forrásait ahhoz, hogy meg tudjuk gyõzõdni a biztonságosságáról.

  Ez a fal lehet például egy felhasználói azonosító. A security(7) és named(8) man oldalakon is ezt a definíciót találjuk meg.

  Vegyük például az `ntalk` szolgáltatást (lásd inetd(8)). Ezt a szolgáltatást korábban a `root` felhasználó azonosítójával futtatták, de manapság viszont már a `tty` felhasználóval fut. A `tty` felhasználó lényegében egy olyan járóka, amely az `ntalk` szolgáltatás feltörésekor nem engedi, hogy a rendszer többi részéhez is hozzá lehessen férni.

- A valódi gépet utánzó rendszerben futó programot. Ez már egy sokkal kifinomultabb megoldás. Ha ilyenkor valakinek sikerül betörnie a programba, akkor könnyen azt hiheti, hogy sikerült a rendszer többi részét is elérnie, de valójában csak egy szimulált gépen van, és semmilyen valós adatot nem képes módosítani.

  Leggyakrabban ezt úgy szokták elérni, hogy egy könyvtárban létrehoznak egy szimulált környezetet, majd itt futtatják az adott programot a chroot(8) segítségével. (Ekkor az iménti könyvtár lesz a gyökérkönyvtár az adott folyamat számára, nem pedig a rendszer igazi gyökere.)

  Másik szintén gyakori megoldás a használt állományrendszerek írásvédett csatlakoztatása, amely felett aztán kialakítanak a program számára egy látszólag írható réteget. Ilyenkor a program teljesen úgy érzékeli, hogy képes a rendszerben elérhetõ állományokat módosítani, azonban egyedül csak saját maga látja ezeket, a rendszerben futó többi program viszont nem feltétlenül.

  Ezeket a járókákat általában úgy szokták felépíteni, hogy a felhasználók (vagy a támadók) számára teljesen észrevétlenek legyenek.

A UNIX® két alapvetõ járókát valósít meg. Az egyik a futó programok, a másik pedig a felhasználói azonosítók szintjén mûködik.

Futása közben minden UNIX® program teljesen elszigetelt minden más UNIX® programtól, így az egyik nem képes módosítani a másik memóriában tárolt adatait. A Windows®-tól eltérõen, ahol ugyebár az egyik program könnyedén el tudja érni egy másik memóriaterületét, ezért a program nem képesek egymásban kárt tenni.

A UNIX® alatt futó programok mindig egy adott felhasználóhoz tartoznak. Ha ez nem a `root`

felhasználó, akkor azzal lényegében egy tûzfalat hozunk létre a különbözõ felhasználók által birtokolt folyamatok között. A felhasználók azonosítói emellett segítenek a lemezen tárolt adatokat is elszigetelni egymástól.

13.2. Mi az a biztonsági szint (securelevel)?

A biztonsági szintek egy rendszermagon belül megvalósított védelmi módszert képviselnek. A pozitív értékû biztonsági szintek esetén a rendszermag korlátoz bizonyos feladatokat, amelyeket ilyenkor még a rendszeradminisztrátor (vagyis a `root` felhasználó) sem képes elvégezni. Az írás pillanatában a biztonsági szintek, több más dolog mellett, a következõk szabályozására alkalmasak:

- a különbözõ állományjelzõk, például az `schg` (a "system immutable" jelzés) törlése;
- a rendszermag memóriájának elérése a /dev/mem és /dev/kmem eszközökön keresztül;
- a rendszermag moduljainak betöltése;
- a tûzfal szabályainak módosítása.

A jelenleg futó rendszer biztonsági szintjét a következõ parancs segítségével lehet lekérdezni:

```
# sysctl kern.securelevel
```

A parancs eredménye az adott sysctl(8) változó (vagyis esetünkben a `kern.securelevel`) és annak értéke lesz, amely egy szám. Ez utóbbi adja meg a biztonsági szint aktuális értékét. Amennyiben ez pozitív (vagyis nullánál nagyobb), akkor érvényben vannak a biztonsági szintekhez kapcsolódó bizonyos korlátozások.

Egy mûködõ rendszer biztonsági szintjét nem lehet csökkenteni, hiszen ezzel tulajdonképpen hatástalanná tennénk. Ha olyan feladatot akarunk végrehajtani, amely nem pozitív biztonsági szintet igényel (például az alaprendszer frissítése vagy a dátum átállítása), akkor ahhoz elõször módosítanunk kell az /etc/rc.conf állományt (lásd `kern_securelevel` és `kern_securelevel_enable` változók), majd újraindítani a rendszert.

A biztonsági szintekkel és rájuk vonatkozó információkkal kapcsolatban olvassuk el az init(8) man oldalt.

A biztonsági szintek nem feltétlenül jelentenek minden problémára tökéletes megoldást. Rengeteg ismert hátulütõvel rendelkeznek, és gyakran a biztonság hamis érzetét keltik.

Ezzel kapcsolatban az egyik legnagyobb gond, hogy csak abban az esetben mûködik rendesen a rendszer, ha a rendszerindítás során a biztonsági szintek beállításáig minden állományt levédünk. Ha a támadó képes lefuttatni a saját programját még a biztonsági szint beállítása elõtt (amely viszont elég késõn történik meg, hiszen a rendszerindítás során számos olyan dolog feladat van, amely nem végezhetõ el magasabb biztonsági szinteken), akkor azzal az egész védelmi módszer hatástalanítható. Habár a rendszerindítás folyamán felhasznált állományok biztonságba helyezése technikailag egyáltalán nem lehetetlen,

nehezebbé válik tõle a rendszer karbantartása, mivel ilyenkor az egész rendszert át kell állítanunk legalább egyfelhasználós módba és úgy módosítani a konfigurációs állományokat.

Ezt és az ehhez hasonló problémák gyakran felmerülnek a levelezési listákon, különösen a FreeBSD security levelezési lista archívumaiban. Ezen a funkción keresztül nézhetünk után a téma részletesebb tárgyalásának. Néhányan reménykednek, hogy a biztonsági szinteket hamarosan leváltja valami sokkal finomabb beállítási lehetõségekkel rendelkezõ megoldás, azonban a dolgok még eléggé homályosak ebbõl a szempontból.

Figyelmeztettünk mindenkit!

13.3. A BIND (named) különféle nagyobb sorszámú portokat használ. Miért?

A BIND a kimenõ kérésekhez véletlenszerûen kiválaszt egy nagyobb sorszámú portot. A legújabb változataiban már minden egyes kéréshez külön véletlenszerûen keres új UDP portot. Ez bizonyos hálózati konfigurációk esetén problémákhoz vezethet, különösen olyankor, amikor a beérkezõ UDP csomagokat egy tûzfal megállítja. A tûzfalak által blokkolt porttartományok használatát az `avoid-v4-udp-ports` vagy az `avoid-v6-udp-ports` beállítással tilthatjuk le a program számára.

Ha ezt a portot (mint például az 53) az /etc/namedb/named.conf állományban a `query-source` vagy a `query-source-v6` beállításokkal adjuk meg explicit módon, akkor a program nem fogja véletlenszerûen váltogatni a portokat. Határozottan javasoljuk, hogy ezekkel az opciókkal ne adjunk meg elõre rögzített portokat.

Mindenesetre örülünk, hogy ezt is valaki megkérdezte! Hiába, nem árt néha nézegetni a sockstat(1) kimenetét és észrevenni benne néhány furcsaságot.

13.4. A sendmail a szabványos 25-ös port mellett az 587-es portot használja! Miért?

A sendmail újabb verzióiban felfedezhetõ levélküldési lehetõségek az 587-es portot használják. Jelenleg ezt még nem sokan használják ki, de növekszik a népszerûsége.

13.5. Kié az a nullás felhasználói azonosítójú toor fiók? Betörtek a gépre?

Ne aggódjunk! A `toor` egy "alternatív" rendszergazdai hozzáférés (a "toor" a "root" visszafelé). Korábban csak a bash(1) parancsértelmezõ telepítésekor jött létre, azonban manapság már alapértelmezés szerint létrejön. A nem szabványos parancsértelmezõk számára találták ki, így nem a `root` alapértelmezett parancsértelmezõjét kell megváltoztatnunk. Ez különösen olyan parancsértelmezõk esetén fontos, amelyek nem részei az alaprendszernek (például a portként vagy csomagként telepített parancsértelmezõk esetén) és ezért a /usr/local/bin könyvtárba fognak

kerülni. Ez a könyvtár alapértelmezés szerint azonban egy külön állományrendszeren található. Ha a `root` parancsértelmezõje viszont a /usr/local/bin könyvtárban lenne, miközben a /usr (vagy bármelyik más állományrendszer, amely az imént említett könyvtárat tartalmazza) nem csatlakoztatható valamilyen oknál fogva, akkor a `root` nem lenne képes bejelentkezni és kijavítani a problémát. (Noha amikor újraindítjuk a rendszerünket egyfelhasználós módban, akkor a rendszer rá fog kérdezni, hogy melyik parancsértelmezõt akarjuk használni.)

Egyesek nem szabványos parancsértelmezõn keresztül a `toor` felhasználóval végzik el a `root` mindennapi teendõit, így a szabványos parancsértelmezõt csak a vészhelyzetekre tartogatják. Alapértelmezés szerint a `toor` felhasználóval nem tudunk bejelentkezni, mivel nincs jelszava, ezért ha használni akarjuk, akkor elõször jelentkezzünk be a `root` felhasználóval, majd állítsunk be neki egy jelszót.

13.6. A suidperl parancs miért nem mûködik rendesen?

Biztonsági okokból a `suidperl` parancs alapértelmezés szerint nem kerül telepítésre. Ha forrásból frissítjük rendszerünket és azt szeretnénk, hogy ennek során a `suidperl` is lefordúljon, akkor a `perl` fordításának megkezdése elõtt vegyük fel a `ENABLE_SUIDPERL=true` sort az /etc/make.conf állományba.

Chapter 14. PPP

14.1. Nem mûködik a ppp(8). Mit lehet a gond?

Elsõként mindenképpen olvassuk el a ppp(8) man oldalt és a kézikönyv PPP-vel foglalkozó részét. A következõ paranccsal engedélyezzük a naplózást:

```
set log Phase Chat Connect Carrier lcp ipcp ccp command
```

Ezt a parancsot a ppp(8) parancssorában vagy az /etc/ppp/ppp.conf konfigurációs állományban kell megadnunk (leginkább a `default` szakasz elejére érdemes betennünk). Gondoskodjunk róla, hogy az /etc/syslog.conf állomány (lásd syslog.conf(5)) tartalmazza az alábbi sort, illetve az /var/log/ppp.log állomány létezzen:

```
!ppp
*.*       /var/log/ppp.log
```

A napló segítségével már több mindent ki tudunk deríteni a ppp(8) mûködésérõl. Ne aggódjunk, ha nem értünk belõle semmit. Kérjünk segítséget másoktól, nekik minden bizonnyal segíteni fog a probléma felderítésében.

14.2. A ppp(8) miért bontja a vonalat, amikor elindul?

Ilyen általában azért történik, mert nem tudta feloldani a hálózati nevet. Ezt a legkönnyebben úgy tudjuk orvosolni, ha az /etc/host.conf állományba elõre rakjuk a `hosts` sort, így a névfeloldó elõször az /etc/hosts állománnyal fog próbálkozni. Ezután a /etc/hosts állományba vegyük fel a helyi gépet. Ha nincs helyi hálózatunk, akkor így írjuk át a `localhost` sort:

```
127.0.0.1       ize.minta.com ize localhost
```

Minden más esetben egyszerûen csak vegyünk fel egy újabb bejegyzést a gépünkhöz. Ennek pontosabb részleteivel kapcsolatban nézzük meg a megfelelõ man oldalakat.

Ha mindent jól csináltunk, akkor a `ping -c1 hostname` parancs hiba nélkül tér vissza.

14.3. A ppp(8) miért nem tárcsáz -auto módban?

Elõször is ellenõrizzük, hogy létezik az alapértelmezett útvonal. A `netstat -rn` parancs (lásd netstat(1)) kiadása után nagyjából a következõ két bejegyzést kell látnunk:

```
Destination        Gateway            Flags     Refs     Use     Netif Expire
default            10.0.0.2           UGSc        0        0      tun0
10.0.0.2           10.0.0.1           UH          0        0      tun0
```

Feltételezzük, hogy a kézikönyvbõl, a man oldalról vagy ppp.conf.sample állományból másoltuk ki ezeket a címeket. Ha nincs alapértelmezett útvonalunk, akkor annak az lehet az oka, hogy a ppp.conf állományba elfelejtettük felvenni a `HISADDR` kulcsszót.

Az alapértelmezett útvonal hiányának másik oka lehet még, ha az /etc/rc.conf állományban (lásd rc.conf(5)) beállítottunk egy alapértelmezett átjárót, de elfelejtettük az ppp.conf állományba betenni a következõ sort:

```
delete ALL
```

Ebben az esetben menjünk vissza a kézikönyv A rendszer végsõ beállítása címû részéhez.

14.4. Mit jelent a No route to host hibaüzenet?

Általában azért jelentkezik, mert az /etc/ppp/ppp.linkup állományban nem adtuk meg az alábbiakat:

```
MYADDR:
  delete ALL
  add 0 0 HISADDR
```

Erre csak akkor van szükségünk, ha dinamikus IP-címünk van, vagy nem ismerjük az átjáró címét. Ha az interaktív módot használjuk, akkor ehhez a következõket kell begépelni csomag módba lépés után (a csomag módot a csupa nagybetûs PPP jelzi a parancssorban):

```
delete ALL
add 0 0 HISADDR
```

A kézikönyv A PPP és a dinamikus IP-címek címû részében olvashatunk errõl bõvebben.

14.5. Miért szakad meg a kapcsolat 3 perc után?

A PPP alrendszer alapértelmezett lejárati ideje 3 perc. Ezt a beállítást a következõ sor megadásával tudjuk módosítani:

```
set timeout NNN
```

ahol az *NNN* másodpercekben megadja a kapcsolat lezárása elõtti inaktivitás maximális idejét. Ha az *NNN* értéke nulla, akkor a kapcsolat idõtúllépés miatt soha nem fog magától megszakadni. Ezt a parancsot a ppp.conf állományba tudjuk felvenni, vagy interaktív módban a parancssorban gépelhetjük be. Emellett menet közben is állítani tudjuk ezt az értéket, ha a ppp szerverre a telnet(1) vagy a pppctl(8) segítségével rácsatlakozunk. Errõl a ppp(8) man oldal ad részletesebb tájékoztatást.

14.6. A kapcsolat miért szakad meg nagyobb terhelést alatt?

Ha beállítottuk a Link Quality Reporting (LQR) használatát, akkor elõfordulhat, hogy túlságosan sok csomag veszik el a gépünk és a másik oldal között. A ppp(8) ezért a vonalat rossznak érzékeli és bontja. A FreeBSD 2.2.5 változata elõtt az LQR alapértelmezés szerint engedélyezett volt. Az LQR így tiltható le:

```
disable lqr
```

14.7. A kapcsolat miért szakad meg véletlenszerûen?

Néha elõfordulhat, hogy a zajos telefonvonal esetén vagy a hívásvárakoztatás használatakor a modem bontja a vonalat, mivel (helytelenül) azt hiszi, hogy nincs kapcsolat.

Manapság a legtöbb modemen általában be lehet valahogy állítani, hogy mennyire legyenek elnézõek a kapcsolat ideiglenes megszakadásával szemben. Például egy U.S. Robotics® Sportster® esetén ezt tizedmásodpercekben mérik az `S10` regiszter segítségével. A modemünk ilyenkor tehát úgy tehetõ sokkal toleránsabbá, ha a következõ hívási beállítást adjuk:

```
set dial "...... ATS10=10 OK ......"
```

További részleteket a modem kézikönyvébõl tudhatunk meg.

14.8. A kapcsolat miért fullad le véletlenszerûen?

Sokan tapasztalják, hogy a kapcsolat minden különösebb magyarázat nélkül lefullad. Ilyenkor elsõként azt érdemes tisztázni, hogy az összeköttetés melyik oldalán történt a vonal bontása.

Ha belsõ modemet használunk, akkor próbáljuk meg a ping(8) paranccsal ellenõrizni, hogy a modem TD lámpája villog-e az adatok küldésekor. Amennyiben igen (miközben az RD lámpa viszont nem), akkor a gond a vonal másik végén lesz. Ha viszont a TD nem villog, akkor a probléma a mi oldalunkon áll fenn. A belsõ modemek esetében a ppp.conf állományban a `set server` parancsot is érdemes megadnunk, így amikor a kapcsolat leállását tapasztaljuk, a pppctl(8) segítségével rá tudunk csatlakozni a ppp(8) démonra. Ha a hálózati kapcsolat ekkor hirtelen erõre kapna (mivel rácsatlakoztunk kívülrõl) vagy egyáltalán nem tudunk csatlakozni (feltételezve, hogy a `set socket` parancs sikeresen lefutott az induláskor), akkor a probléma még mindig nálunk lesz. Ha viszont sikerül csatlakoznunk és a vonallal még mindig gondok vannak, akkor próbáljuk a `set log local async` parancs használatával engedélyezni a helyi aszinkron naplózást, majd egy másik konzolból a ping(8) parancs segítségével kezdjük el használni az összeköttetést. Az aszinkron naplózás jelezni fogja, ha sikerül adatokat átvinni és fogadni a kapcsolaton keresztül. Ha ilyenkor nem látunk visszafele érkezõ adatokat, akkor az arra utal, hogy a gond a vonal távoli végén van.

Miután sikeresen kiderítettük, hogy az adott probléma helyi vagy távoli, két lehetõségünk van:

- Amennyiben távoli, olvassuk el a A vonal túlsó végérõl nem érkezik válasz. Mi lehet tenni? válaszát.
- Amennyiben helyi, olvassuk el a A ppp(8) teljesen megállt. Mi lehet tenni? válaszát.

14.9. A vonal túlsó végérõl nem érkezik válasz. Mi lehet tenni?

Ezzel szemben nagyon keveset tudunk mi, felhasználók tenni. A legtöbb internetszolgáltató egyszerûen nem hajlandó segítséget nyújtani abban az esetben, ha nem valamelyik Microsoft® operációs rendszert használjuk. A ppp.conf állományunkban a `enable lqr` sor megadásával engedélyezni tudjuk a ppp(8) számára, hogy észlelhesse a távoli hibákat és bontsa a vonalat, de ez a vizsgálat viszonylag idõigényes és ennélfogva nem túlságosan hasznos. A szolgáltatónknak pedig ne nagyon emlegessük, hogy felhasználói PPP-t futtatunk.

Elõször próbáljunk meg letiltani mindenféle tömörítést a következõ sor megadásával:

```
disable pred1 deflate deflate24 protocomp acfcomp shortseq vj
deny pred1 deflate deflate24 protocomp acfcomp shortseq vj
```

Kapcsolódjunk újra és ellenõrizzük, hogy továbbra is mûködõképes a kapcsolat. Ha ennek hatására javul a helyzet vagy a probléma teljesen megoldódik, akkor a beállítások egyenkénti próbálgatásával keressük meg, hogy melyik okozta a gondot. Ez már elegendõ lesz ahhoz, hogy komolyabban felvegyük a kapcsolatot a szolgáltatónkkal (habár ebbõl gyorsan ki fog derülni, hogy nem Microsoft® terméket használunk).

Mielõtt szólnánk a szolgáltatónknak, a gépünkön engedélyezzük az aszinkron naplózást és várjuk meg, amíg a kapcsolat újra megszakad. Erre nem árt felkészülnünk, mert viszonylag sok tárhelyet igényel. Innen majd a portról utoljára olvasott adat lesz a lényeges. Ez általában szöveges adat és akár a probléma konkrét okára is utalhat (`Memory fault`, `Core dumped`?).

Ha segítõkész szolgáltatót választottuk, akkor a naplózást akár az õ oldalunkon is engedélyezhetjük, így amikor a vonal megszakad, az õ szemszögükbõl is képesek leszünk elemezni a problémát. Ilyen esetben nyugodtan küldjünk egy levelet Brian Somers <brian@FreeBSD.org> címére vagy kérjük meg a szolgáltatónkat, hogy közvetlenül vele tárgyaljon.

14.10. A ppp(8) teljesen megállt. Mi lehet tenni?

A legjobban úgy járunk, ha a ppp(8) programot nyomkövetési információkkal fordítjuk újra, majd a gdb(1) segítségével lekérünk egy hívási láncot az éppen megakadt ppp példánytól. A ppp alkalmazást a következõ parancsokkal tudjuk úgy újrafordítani, hogy tartalmazza a kívánt információkat:

```
# gdb ppp `pgrep ppp`
```

Ezt követõen a gdb parancssorában a `bt` és `where` parancsok segítségével hozzá tudunk jutni a hívási

lánchoz. Mentsük el valahova a gdb által kinyert adatokat, majd a `detach` paranccsal váljunk le a futó programról és a `quit` begépelésével lépjünk ki a gdb programból.

Végezetül az elmentett eredményeket küldjük el Brian Somers <brian@FreeBSD.org> címére.

14.11. Miért nem történik semmi a Login OK! üzenet után?

A FreeBSD 2.2.5 elõtti kiadásaiban a ppp(8) az összeköttetés létrejötte után megvárta, hogy a távoli pont kezdeményezze a kapcsolatvezérlõ protokoll (Line Control Protocol, LCP) használatát. Sok szolgáltató azonban nem csinál ilyet, ehelyett inkább a klienstõl várják mindezt. Az LCP kezdeményezését így kényszeríthetjük ki a ppp(8) használata során:

```
set openmode active
```

Általában semmilyen gond nem származik abból, ha a mind a két oldal kezdeményez, így az `openmode` alapértelmezés szerint `active` értékû. A következõ szakaszban azonban bemutatjuk mikor *gondot okoz* a használata.

14.12. Folyamatosan Magic is same hibák jelennek meg. Ez mire utal?

Csatlakozás után idõnként elõfordulhat, hogy `magic is the same` hibaüzeneteket látunk a naplóban. Ezek az üzenetek bizonyos esetekben teljesen ártalmatlanok, máskor viszont akár komolyabb problémákat is jelezhet. A legtöbb PPP implementáció nem él túl egy ilyen hibát, és még ha látszólag létre is jön ilyenkor a kapcsolat, folyamatosan konfigurációs kérések és válaszok jönnek-mennek a naplóban egészen addig, amíg a ppp(8) végül fel nem adja és lezárja a kapcsolatot.

Ez általában olyan szervereken jelenik meg, ahol nem elég gyorsak a lemezek és minden kapcsolathoz elindítanak egy getty(8) és a bejelentkezéskor vagy azt következõ elindítják a ppp(8) programot. Egyes visszajelzések szerint ilyen egyébként gyakran elõfordul a slirp használatakor. A problémát egyébként a getty(8) és a ppp(8) indítása között eltelt idõ okozza, amikor a kliens oldalán futó ppp(8) elkezdi küldeni a kapcsolatvezérlõ (Line Control Protocol, LCP) csomagokat. Mivel ilyenkor az ECHO még mindig aktív a szerver adott portján, a kliens ppp(8) a saját csomagjainak "tükrözõdését" fogja látni.

Az LCP beállításának része az összeköttetés két oldalán egy-egy bûvös szám ("magic number") megállapítása, amellyel ezután észlelhetõek az ilyen "tükrözõdések". A protokoll szerint amikor a két pont megpróbálja ugyanazt a bûvös számot használni, akkor visszautasítja (NAK jelzést küld) és egy másikat választ. Ha ilyenkor még a szerver portján aktív az ECHO, akkor a kliens oldali ppp(8) azt tapasztalja, hogy elkezd LCP csomagokat küldeni, majd mivel ugyanazt kapja vissza, erre egy NAK jelzést válaszol. Ugyanígy látja magát a NAK jelzést (aminek hatására a ppp(8) megváltoztatja a bûvös számát) is. Ennek eredményeképpen hirtelen nagy mennyiségû bûvösszám-váltás keletkezik, ami pedig szépen felhalmozódik a szerver terminálpufferében. Ahogy a ppp(8) végre elindul a szerveren, elönti ez a rengeteg információ, aminek alapján sikertelennek ítéli meg az LCP

beállítását és feladja a további próbálkozást. Eközben a kliens számára megszûnnek a visszaverõdõ csomagok és csak annyit lát, hogy a szerver bontja a kapcsolatot.

Ezt úgy tudjuk elkerülni, ha a ppp.conf állományban a távoli pontra bízzuk az beállítás kezdeményezését:

```
set openmode passive
```

Ennek hatására a ppp(8) megvárja, hogy a szerver kezdeményezze az LCP beállítását. Egyes szerverek azonban sosem teszik meg ezt. Ilyenkor valami ilyesmit tudunk tenni:

```
set openmode active 3
```

Így a ppp(8) 3 másodpercig passzív marad, majd csak ezután kezd el LCP kérésket küldeni. Ha a távoli pont eközben küld valamilyen kérést, az ppp(8) azonnal válaszol rá és nem várja végig a 3 másodperces idõtartamot.

14.13. Az LCP beállítása egészen a kapcsolat befejezõdéséig folytatódik. Mi lehet a probléma?

A ppp(8) programban jelenleg van egy olyan hibásan implementált jellemzõ, ahol az LCP, CCP és IPCP válaszokat nem társítja az eredeti kérésekhez. Ennek következményeképpen, ha az egyik PPP implementáció 6 másodperccel lassabb a másik oldalnál, akkor az még két további LCP konfigurációs kérést is küld, ami viszont végzetesnek bizonyul.

Vegyünk például két implementációt, az `A` és a `B` pontokat. Az `A` már közvetlenül a csatlakozás után LCP kéréseket kezd el küldeni, miközben a `B` csak 7 másodperc múlva tud elindulni. Mire végre a `B` pont is elindul, addigra az `A` már kiküldött 3 LCP kérést. Most feltételezzük, hogy nincs ECHO, máskülönben az elõzõ szakaszban leírt, bûvös számokkal kapcsolatos problémába ütköznénk. A `B` ekkor tehát küld egy kérést, majd nyugtázza az `A` ponttól kapott korábbi kérést. Ennek hatására az `A` pont OPENED állapotba megy át, újra küld és nyugtázza az elõzõ kérést `B` felé. Eközben a `B` további két nyugtázást küld az `A` pontról kapott további két kérésre, a `B` indulása elõttrõl. A `B` ekkor megkapja az `A` elsõ nyugtáját és átvált OPENED állapotba. Az `A` ekkor megkapja a második nyugtát a `B` ponttól és visszavált REQ-SENT állapotba, majd az RFC szerint elküld (elõre) egy újabb kérést. Ekkor megkapja a harmadik nyugtát és OPENED állapotba vált. Eközben a `B` megkapja elõre küldött kérést a `A` ponttól, amelynek hatására ACK-SENT állapotba vált vissza, és az RFC szerint ismét küld egy (második) kérést és egy nyugtázást. Az `A` erre megkapja a kérést, visszavált REQ-SENT állapotban és küld egy újabb kérést. Ekkor közvetlenül megkapja a rákövetkezõ nyugtázást és átvált OPENED állapotba.

Ez egészen addig folytatódik, amíg az egyik oldal rá nem eszmél, hogy ennek nincs túlságosan sok értelme és feladja a próbálkozást.

Ez legkönnyebben úgy kerülhetõ el, ha ilyenkor az egyik oldalt `passive` típusúra állítjuk, vagyis az egyik oldalon várunk egy keveset a beállítás kezdeményezésére. Ezt a következõ paranccsal lehet megoldani:

```
set openmode passive
```

Óvatosan bánjunk ezzel a paraméterrel! A beállítás kezdeményezésének várakoztatási idejét a következõ paraméterrel tudjuk megadni:

```
set stopped N
```

Használhatjuk viszont ezt a parancsot is (ahol *N* adja meg, hogy mennyi másodperc teljen el a beállítás megkezdése elõtt):

```
set openmode active N
```

Az ezzel kapcsolatos további részleteket a man oldalon olvashatjuk.

14.14. Miért akad meg a ppp(8), ha egy külsõ parancsot adunk ki alatta?

A `shell` vagy `!` parancsok végrehajtásakor a ppp(8) elindít egy parancsértelmezõt (illetve ha paramétereket is adtunk meg, akkor a ppp(8) átadja azokat is), majd megvárja annak befejezõdését. Ha a parancs futtatása közben éppen egy PPP kapcsolatot akartunk használni, akkor erre az idõre az elõbbiek miatt látszólag meg fog állni. Ez tehát azért történik, mert a ppp(8) megvárja a parancs lefutását.

Ha nem akarjuk megvárni a parancs befejezõdését, akkor inkább használjuk a `!bg` parancsot. Ennek hatására az adott parancs a háttérben fog lefutni és a ppp(8) képes lesz folyamatosan szemmel tartani az összeköttetést.

14.15. A ppp(8) null-modem kábel használatakor miért nem lép ki soha?

A ppp(8) ilyen esetekben nem képes magától megállapítani, hogy mikor bontották a vonalat. Ennek oka a tûk null-modem kábelben kiosztott szerepében keresendõ. Amikor ilyen típusú kapcsolattal dolgozunk, a következõ sor megadásával ne felejtsük el engedélyezni az LQR használatát:

```
enable lqr
```

Ha a távoli pont LQR csomagokat küld, akkor a ppp(8) alapértelmezés szerint fogadja azokat.

14.16. A ppp(8) miért tárcsáz látszólag minden különösebb ok nélkül -auto módban?

Amennyiben a ppp(8) szándékainkkal szemben váratlanul kezdene el tárcsázni, akkor keressük meg kiváltó okát és használjunk hívási szûrést (Dial filter, dfilter) ennek megelõzésére.

A tárcsázás okát a következõ sor használatával tudjuk kideríteni:

```
set log +tcp/ip
```

Ennek hatására a kapcsolaton keresztüláramló összes forgalmat naplózni fogjuk. Így a legközelebb, amikor a vonal hirtelen aktív lesz, a hozzá tartozó idõbélyegek alapján könnyen elõ tudjuk keresni, hogy pontosan miért is történt.

Az automatikus tárcsázást bizonyos esetekben le tudjuk tiltani. Ez általában egy olyan probléma, amely a névfeloldások miatt keletkezik. Úgy tudjuk megakadályozni, hogy a névfeloldások felépítsék a kapcsolatot (ami viszont *nem* gátolja abban a ppp(8) programot, hogy egy már meglevõ kapcsolaton keresztül küldjön ilyen csomagokat), ha az alábbi beállításokat adjuk meg:

```
set dfilter 1 deny udp src eq 53
set dfilter 2 deny udp dst eq 53
set dfilter 3 permit 0/0 0/0
```

Ezek az értékek nem minden esetben megfelelõek számunkra, hiszen ezzel együtt az igény szerinti tárcsázás kényelmét is szûkítjük, mivel a legtöbb program közvetlenül névfeloldással kezd, mielõtt komolyabb hálózati forgalmat bonyolítana le.

A névfeloldás esetében igyekezzünk kideríteni, hogy pontosan melyik program is próbál hálózati neveket feloldatni. Az esetek többségében valószínûleg a sendmail(8) lesz a bûnös. Amennyiben ez a helyzet, akkor az sendmail démonnak a saját konfigurációs állományában kell beállítanunk, hogy ne oldasson fel hálózati neveket. Az érintett konfigurációs állomány módosításának pontos részleteirõl a kézikönyv Levelezés betárcsázós kapcsolattal címû szakszában olvashatunk bõvebben. Továbbá az .mc állományunkba a következõ sort is érdemes felvennünk:

```
define(`confDELIVERY_MODE', `d')dnl
```

Ezzel a sendmail beindításáig mindent egy sorban fog eltárolni (általában a sendmail démont a `-bd -q30m` paraméterekkel szokták meghívni, ami arra utasítja, hogy 30 percenként dolgozza fel a sorát) vagy amíg a `sendmail -q` parancs le nem fut (például a ppp.linkup állományból).

14.17. Mit jelentenek a CCP hibák?

A naplóban folyamatosan a következõ üzeneteket lehet látni:

```
CCP: CcpSendConfigReq
CCP: Received Terminate Ack (1) state = Req-Sent (6)
```

Ilyenek azért keletkeznek, mert a ppp(8) a `Predictor1` tömörítési eljárást próbálja meg beállítani, azonban a távoli pont egyáltalán semmilyen tömörítést nem akar használni. Az ilyen üzenetek többnyire ártalmatlanok, de ha el akarjuk tüntetni ezeket, akkor próbáljuk meg a következõ módon kikapcsolni a `Predictor1` tömörítés használatát:

```
disable pred1
```

14.18. A ppp(8) miért nem naplózza a kapcsolat sebességét?

A modemmel végzett teljes "beszélgetés" szövegének rögzítéséhez a következõket kell engedélyezni:

```
set log +connect
```

Ennek eredményeképpen a ppp(8) egészen az utolsóként lekért karakterláncig naplóz mindent.

Ha PAP vagy CHAP hitelesítést használunk (ezért a `CONNECT` parancs kiadása után már nincs semmi "mondanivalónk" a hívószkriptben, tehát nincs `set login` szkript), és szeretnénk látni a csatlakozási sebességet, ne felejtsük el utasítani a ppp(8) programot, hogy a teljes `CONNECT` sort kérje le, valahogy így:

```
set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 4 \
  \"\" ATZ OK-ATZ-OK ATDT\\T TIMEOUT 60 CONNECT \\c \\n"
```

Itt most megkapjuk a `CONNECT` sort, ezután nem küldünk semmit, majd várunk egy soremelést, aminek hatására a ppp(8) arra kényszerül, hogy a teljes `CONNECT` választ beolvassa.

14.19. A ppp(8) miért hagyja figyelmen kívül a \ karaktereket a szkriptekben?

A ppp a konfigurációs állományokból minden sort külön beolvas, ezért a `set phone "123 456 789"` és hozzá hasonló karakterláncok esetén képes felismerni, hogy a megadott számok valójában *egyetlen* paramétert formáznak. A `"` megadásához a visszaper karaktert (`\`) kell használnunk.

Amikor tárcsázásért felelõs értelmezõ beolvassa az egyes paramétereket, újraértelmezi ezeket olyan speciális helyettesítési szekvenciák után kutatva, mint például a `\P` vagy `\T` (részletesebben lásd a man oldalon). A kettõs elemzés miatt nekünk is a megfelelõ számban kell megadnunk ezeket a helyettesítendõ karaktereket.

Ha tehát egy `\` karaktert szeretnénk átküldeni a modemünknek, akkor nagyjából valami ilyesmit

kellene írnunk:

```
set dial "\"\" ATZ OK-ATZ-OK AT\\\\X OK"
```

Ennek az eredménye a következõ lesz:

```
ATZ
OK
AT\X
OK
```

Vagy:

```
set phone 1234567
set dial "\"\" ATZ OK ATDT\\T"
```

Ez pedig a következõ szekvenciát adja:

```
ATZ
OK
ATDT1234567
```

14.20. A ppp(8) miért küld Segmentation Fault hibát, miközben nem is keletkezik ppp.core állomány?

A ppp (vagy más hasonló program) elméletileg soha nem hoz létre .core állományt. Mivel a ppp(8) tulajdonképpen a nullás felhasználói azonosítóval fut, az operációs rendszer soha nem fogja a ppp(8) memórialenyomatát leállítása elõtt a lemezre menteni. Ha viszont ppp(8) mûködése valóban leáll egy szegmentációs hiba vagy bármilyen más .core állományt eredményezõ jelzés miatt, *és* valóban a legfrissebb változatát használjuk (lásd a fejezet elejét), akkor a következõt tehetjük:

```
# cd /usr/src/usr.sbin/ppp
# echo STRIP= >> /etc/make.conf
# echo CFLAGS+=-g >> /etc/make.conf
# make install clean
```

A fenti parancsokkal telepíteni tudjuk a ppp(8) egy nyomonkövethetõ változatát. A ppp(8) futtatásához `root` felhasználónak kell lennünk, mivel minden korábbi engedélyét felülírtuk az elõbbiek során. A ppp(8) indításakor ne felejtsük el megjegyezni pontosan az aktuális könyvtárat sem.

Innentõl kezdve, amikor a ppp(8) kap egy szegmentációs hibára vonatkozó jelzést, létre fog hozni egy ppp.core nevû állományt. Ennek birtokában a következõt kell csinálnunk:

```
% su
# gdb /usr/sbin/ppp ppp.core
(gdb) bt
..
(gdb) f 0
..
(gdb) i args
..
(gdb) l
..
```

Az így beszerzett információkat mellékelve nagyobb valószínûséggel kaphatunk választ az ezzel kapcsolatos kérdésünkre.

Ha járatosak vagyunk a gdb(1) használatában, akkor a .core állományban további részletek és információk utáni is kutathatunk, például mi okozta a hibát, milyen változóknak ekkor milyen értékei voltak stb.

14.21. Miért nem csatlakozik soha az a program, amely a hívást kezdeményezte -auto módban?

Ez korábban egy ismert probléma volt a ppp(8) használatával kapcsolatban, amikor dinamikus helyi IP-címet akart beállítani `-auto` módban. Ez a hiba az újabb változatokban már nem nincs meg (a man oldalon keressünk rá az `iface` részre).

A gondot az okozta, hogy amikor a tárcsázást elindító program meghívja a connect(2) rendszerhívást, akkor a tun(4) interfészhez tartozó IP-cím a végpontot képviselõ sockethez társul. A rendszermag létrehozza az elsõ kimenõ csomagot és kiírja a tun(4) eszközre. A ppp(8) ekkor beolvassa a csomagot és felépíti a kapcsolatot. Ha a ppp(8) dinamikus IP-cím kiosztásának eredményeképpen ilyenkor az interfész címe megváltozik, akkor azzal egyidõben az eredeti socket végpont érvénytelenné válik. Így a távoli végpont felé küldött további csomagok általában eldobódnak. Ha valahogy mégis eljutnának a céljukhoz, a válasz már semmiképpen sem érkezhet meg, mivel a küldéshez használt IP-címnek már nem az adott gép a tulajdonosa.

Számos elméleti megközelítés létezik az imént felvázolt probléma megoldására. A legszebb az lenne, ha a távoli pont lehetõség szerint a korábban használt IP-címet osztaná ki újra. A ppp(8) jelenlegi változata pontosan ugyanezt teszi, viszont a legtöbbi implementáció már nem.

Részünkrõl az bizonyulna a legegyszerûbb megoldásnak, ha a tun(4) intefész IP-címe egyáltalán nem változhatna meg, hanem helyette menet közben az összes kimenõ csomag, köztük természetesen a forrás IP-címe az interfész IP-címérõl az idõközben beállított IP-címre változna. Ez lényegében az, amit a ppp(8) legújabb változataiban felbukkanó `iface-alias` opció is csinál (a libalias(3) és a ppp(8) `-nat` kapcsolója segítségével): karbantartja az összes korábban használt interfész címét és átfordítja ezeket az utoljára beállított címre.

A másik (és valószínûleg a sokkal megbízhatóbb) lehetõség egy olyan rendszerhívás implementálása lenne, amely képes az összes használatban levõ socketet egyik IP-címrõl a másik IP-

címre átállítani. A ppp(8) ekkor fel tudná használni ezt arra, hogy módosítsa az összes addig futó program socketjét az új IP-cím beállításakor. Ugyanezzel a rendszerhívással a DHCP kliensek is képesek lennének átállítani a socketjeiket.

Lehetõségünk van még IP-cím nélkül is létrehozni interfészeket. A kimenõ csomagok ekkor a `255.255.255.255` IP-címet használnák egészen addig, amíg az elsõ `SIOCAIFADDR` ioctl(2) rendszerhívás le nem zajlik. A ppp(8) feladata ilyenkor a forrás IP-cím megváltoztatása, de ha ez `255.255.255.255`, akkor egyedül csak az IP-címnek és az ellenõrzõösszegnek kell megváltoznia. Ez viszont már valamilyen mértékben trükközést a rendszermagon belül, mivel így könnyen tudunk csomagokat küldeni egy rosszul beállított interfészre is, feltételezve, hogy valamilyen módon képesek vagyunk ilyeneket visszamenõleg helyreállítani.

14.22. A legtöbb játék miért nem mûködik a -nat kapcsoló megadásával?

A játékok és a hozzájuk hasonló alkalmazások általában azért nem mûködnek, amikor a libalias(3) könyvtárat használjuk, mert a távoli gép megpróbál kapcsolódni a belsõ hálózatunkon levõ géphez és kéretlen UDP csomagokat kezd el küldeni neki. A címfordítást végzõ programnak fogalma sincs róla, hogy ezeket a csomagokat egy belsõ gépnek kell továbbküldenie.

Akkor lehetünk biztosak ebben, ha egyedül csak azt a szoftvert indítjuk el, amellyel gondjaink akadtak, majd a vagy az átjáró tun(4) interfészét kezdjük el a tcpdump(1) segítségével, vagy pedig engedélyezzük az átjárón a ppp(8) TCP/IP naplózó funkcióját (`set log +tcp/ip`).

Ahogy elindítjuk a gondokat okozó programot, látnunk kell a csomagjait, ahogy megpróbálnak keresztüljutni az átjárón. Az erre érkezõ válaszolok eldobódnak (ez jelenti a problémát). Jegyezzük fel a csomagokhoz társuló portszámokat és állítsuk el a programot. Csináljuk meg néhányszor ezt a vizsgálatot, így ellenõrizni tudjuk, hogy mindig ugyanazokat a portokat használja-e. Amennyiben úgy tapasztaljuk, hogy igen, akkor az /etc/ppp/ppp.conf állományba a következõ sort kell betenni a megfelelõ helyre a mûködés helyreállításához:

```
nat port protokoll belsõ-gép:port port
```

ahol a *protokoll* lehet `tcp` vagy `udp`, a *belsõ-gép* annak a gépnek a címe, ahova tovább akarjuk küldeni a csomagokat, valamint a *port* a csomagok célportját adja meg.

A fenti parancs megváltoztatása nélkül nem tudjuk ugyanezt a szoftvert más gépeken is használni, és itt azzal most nem is foglalkozunk, hogy miként lehet két belsõ géprõl használni ugyanazt a programot. Mindenesetre annyi biztos, hogy a külvilág felé a belsõ hálózatunk csupán egyetlen gépnek fog látszani.

Ha azt látjuk, hogy az alkalmazás nem mindig ugyanazt a portot használja, akkor három választási lehetõségünk van:

1. Készítsük el a támogatását a libalias(3) függvénykönyvtárhoz. A különbözõ "szélsõséges esetekre" a /usr/src/sys/netinet/libalias/alias_*.c állományokban találhatunk példákat (az alias_ftp.c tökéletes kiindulási alap). Ez általában annyit jelent, hogy beolvasunk bizonyos

ismert kimenõ csomagokat, beazonosítjuk benne azt az utasítást, amelynek hatására a külsõ gép csatlakozni próbál a belsõ géphez egy adott (véletlenszerûen választott) porton, majd beállítunk hozzá egy "útvonalat", így a rákövetkezõ csomagok már tudni fogják, hogy merre menjenek.

Ez ugyan a legnehezebb megoldás, de egyben ez is a legjobb, ráadásul így a szoftver több gépen is mûködtethetõ.

2. Proxy használata. Elõfordulhat, hogy az alkalmazás támogatja a `socks5` protokollt vagy (mint ahogy a `cvsup` is csinálja) rendelkezik "passzív" móddal, és így lehetõleg igyekszik elkerülni azt, hogy a távoli géprõl kapcsolatot próbáljanak meg indítani a helyi gépre.
3. A `nat addr` használatával irányítsunk át mindent a belsõ gépre. Ez viszont egy nagyon durva megközelítés.

14.23. Valaki összeírta már a hasznosabb portok sorszámait?

Egyelõre még nem, de szándékunkban áll összeállítani egy ilyen listát (már amennyiben igény lesz rá). Minden itt szereplõ példában az *belsõ* helyett mindig annak a gépnek a belsõ IP-címét írjuk, amelyrõl játszani akarunk.

- Asheron's Call

  ```
  nat port udp belsõ :65000 65000
  ```

  Manuálisan változtassuk meg a játékon belül a portszámot `65000`-re. Ha a belsõ hálózatunkról több gépen is szeretnénk játszni, akkor mindegyiknek adjuk meg egy egyedi portot (vagyis `65001`, `65002` stb.), majd vegyünk fel mindegyikhez egy-egy `nat port` sort.

- Half Life

  ```
  nat port udp belsõ:27005 27015
  ```

- PCAnywhere 8.0

  ```
  nat port udp belsõ:5632 5632
  ```

  ```
  nat port tcp belsõ:5631 5631
  ```

- Quake

  ```
  nat port udp belsõ:6112 6112
  ```

- Quake 2

  ```
  nat port udp belsõ:27901 27910
  ```

  ```
  nat port udp belsõ:60021 60021
  ```

  ```
  nat port udp belsõ:60040 60040
  ```

- Red Alert

  ```
  nat port udp belsõ:8675 8675
  ```

```
nat port udp belsõ:5009 5009
```

14.24. Mik azok az FCS hibák?

Az FCS jelentése Frame Check Sequence, vagyis az "Adatkeret ellenõrzésének sorozata". Mindegyik PPP csomaghoz tartozik egy ellenõrzõösszeg, amely arról gondoskodik, hogy ugyanaz az adat érkezzen meg, mint amit elküldtek. Amennyiben egy bejövõ csomag FCS értéke érvénytelennek minõsül, a csomag eldobódik és a HDLC FCS számláló értékkel eggyel növekszik. A HDLC hibaszámlálói a `show hdlc` parancs segítségével tekinthetõek meg.

Ha rosszul mûködik az összeköttetés (vagy a soros vonali meghajtónk folyamatosan eldobja a csomagokat), akkor láthatunk helyenként FCS hibákat. Többnyire nem érdemes az ilyenek miatt aggódni, habár ez jelentõs mértékben lassítja a tömörítést végzõ protokollok munkáját. Ha külsõ modemünk van, akkor ne felejtsük el a megfelelõ módon leárnyékolni, mivel ebbõl is származhat a probléma.

Ha a vonal a kapcsolódást követõen szinte azonnal lemerevedik és hirtelen nagy mennyiségû FCS hiba jelentkezik, akkor az arra is utalhat, hogy az összeköttetés nem tisztán 8 bites. Gondoskodjunk róla, hogy a modem ne a szoftveres forgalomirányítást (XON/XOFF) használja. Ha viszont az adatok közvetítéséhez mégis szoftveres forgalomirányítást *kell* használnunk, akkor a `set accmap 0x000a0000` parancs kiadásával jelezzük a ppp(8) felé, hogy a `^Q` és `^S` karaktereket helyettesítse.

Nagy mennyiségû FCS hibát olyan esetekben is tapasztalhatunk, amikor a távoli pont abbahagyta a PPP üzenetek küldését. Ilyenkor javasolt engedélyezni az aszinkron naplózás használatát, aminek segítségével gyorsan meg tudjuk állapítani, hogy a beérkezõ adatok bejelentkezõ vagy shell üzeneteket. Ha a másik oldalon egy shell parancssorát kapjuk meg, akkor a ppp(8) a `close lcp` megadásával a vonal eldobása nélkül leállítható (az utána következõ `term` paranccsal pedig a távoli gépen futó shellre tudunk csatlakozni).

Ha a naplókban látszólag semmi sem indokolja az összeköttetés leállását, próbáljunk meg erre rákérdezni a távoli pont (talán a szolgáltató?) karbantartójánál.

14.25. A Mac OS® és Windows® 98 alól indított kapcsolatok miért állnak le, ha PPPoE fut az átjárón?

A probléma megoldását Michael Wozniak (mwozniak@netcom.ca) adta meg, valamint Dan Flemming (danflemming@mac.com) alkalmazta ugyanezt Macre:

Ennek oka az ún. útválasztási "fekete lyuk". A Mac OS® és a Windows® 98 (de valószínûleg az összes többi Microsoft® operációs rendszer) olyan nagy méretû TCP csomagokat küld, amelyek már nem férnek bele egy PPPoE keretbe (amely mérete Ethernet estén 1500 byte alapértelmezés szerint) *és* beállítja hozzá a darabolás letiltását jelzõ ("do not fragment") bitet (TCP esetén ez alapértelmezett), és a Telco útválasztó pedig nem küldi el a "must fragment" ("darabolni kell") ICMP csomagot a letölteni kívánt oldal szolgáltatója felé. (Másik lehetõség, hogy az útválasztó ugyan küld egy ilyen ICMP csomagot, de ezt a tartalomszolgáltatónál található tûzfal eldobja.) Amikor válaszul a szolgáltató olyan kereteket kezd el küldeni, amelyek nem illeszkednek a PPPoE keresztmetszetébe, a Telco útválasztó egyszerûen eldobja ezeket és a lap nem pedig nem lesz elérhetõ (egyes képek és oldalak esetén elõfordul). Úgy tûnik, ez az alapbeállítás a legtöbb Telco

PPPoE konfiguráció esetében.

Ezt a hibát úgy javíthatjuk, ha a Windows® 95/98 rendszerekben megtalálható `regedit` segítségével felvesszük a következõ regisztrációs bejegyzést:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class\NetTrans\0000\MaxMTU
```

A karakterlánc értéke legyen `1436`, mivel bizonyos ADSL útválasztók állítólag nem képesek ennél nagyobb méretû csomagokat kezelni. Windows® 2000 esetén ezt a beállítást a `Tcpip\Parameters\Interfaces\a hálózati kártya azonosítója\MTU` helyen kell keresni és típusa duplaszó (DWORD).

A Windows® MTU beállításaival kapcsolatban olvassuk el a Microsoft Knowledge Base címén található dokumentumokat: Q158474 - Windows TCPIP Registry Entries és Q120642 - TCPIP & NBT Configuration Parameters for Windows NT® .

Windows® 2000 alatt a regisztrációs adatbázisban érdemes még a `Tcpip\Parameters\Interfaces\a hálózati kártya azonosítója\EnablePMTUBHDetect` duplaszó értékét `1`-re állítani, ahogy arra az imént említett 120642-es Microsoft® dokumentum is hivatkozik.

Sajnos a Mac OS® nem nyújt semmilyen beállítási lehetõséget a TCP/IP beállítások megváltoztatására. Léteznek viszont kereskedelmi termékek, amelyek lehetõvé teszi a felhasználók számára, hogy igényeik szerint módosítsák rendszerük TCP/IP beállításait. A hálózati címfordítást használók keressék meg az MTU beállításaikat és adják meg az `1450` értéket az eredeti `1500` helyett.

A ppp(8) újabb (2.3 vagy afeletti) változatai már tartalmaznak egy `enable tcpmssfixup` parancsot, amellyel az MSS értéke tetszõlegesen átállítható. Ez alapértelmezés szerint engedélyezett. Ha valamiért mégis a ppp(8) egy korábbi változatával kellene dolgoznunk, akkor érdemes megnéznünk net/tcpmssd portot.

14.26. Ezek közül egyik sem használt - segítség! Mit lehetne még tenni?

Ha eddig minden más csõdött mondott, akkor próbáljuk meg elküldeni az összes beszerezhetõ információt, beleértve a konfigurációs állományokat, hogyan indítjuk el a ppp(8) programot, a naplók fontosabb részeit és a `netstat -rn` parancs kimenetét (a csatlakozás elõtt és után) a FreeBSD general questions levelezési lista címére vagy a comp.unix.bsd.freebsd.misc hírcsoportba, és valaki talán majd megmutatja a helyes irányt.

Chapter 15. Soros vonali kommunikáció

Ebben a szakaszban a FreeBSD alatti soros vonali kommunikációval kapcsolatos kérdéseket tárgyaljuk. A PPP és SLIP használatáról a Hálózatok címû részben esik szó.

15.1. Honnan deríthetõ ki, hogy a FreeBSD felismerte a soros portokat a gépben?

Ahogy a FreeBSD rendszermagja az elindulása után azokat a soros portokat fogja keresni, amelyeket a konfigurációs állományban beállítottunk. Figyeljük a rendszer indulása közben megjelenõ üzeneteket vagy adjuk ki a következõ parancsot a rendszer indulásának befejeztével:

```
% dmesg | grep -E "^sio[0-9]"
```

Íme egy példa az iménti parancs kimenetére:

```
sio0: <16550A-compatible COM port> port 0x3f8-0x3ff irq 4 flags 0x10 on acpi0
sio0: type 16550A
sio1: <16550A-compatible COM port> port 0x2f8-0x2ff irq 3 on acpi0
sio1: type 16550A
```

Ezen két soros portot láthatunk. Az elsõ a negyedik megszakítást és a `0x3f8` címet használja és egy 16550A típusú UART chip. A második ugyanolyan chip, de a harmadik megszakítást és a `0x2f8` címet használja. A belsõ modemeket a rendszer úgy kezeli, mintha soros portok lennének, azzal a kivétellel, hogy a modem mindig "kapcsolódik" az adott porthoz.

A GENERIC rendszermag alapértelmezés szerint két soros portot támogat, a példában szereplõ megszakítási- és memóriaértékek felhasználásával. Ha ezek a beállítások nem felelnek meg a rendszerünk számára, esetleg modemet raktunk a gépünkbe vagy a rendszermagban több soros portot is támogatni szeretnénk, akkor nincs más teendõnk, mint ennek megfelelõen megváltoztatni a rendszermag paramétereit. A rendszermag fordításáról szóló rész tárgyalja ennek részleteit.

15.2. Honnan deríthetõ ki, hogy a FreeBSD felismerte a modemkártyát a gépben?

Olvassuk el az elõzõ kérdésre adott választ.

15.3. Hogyan lehet a soros portokat elérni FreeBSD alatt?

A harmadik soros port, a sio2 (lásd sio(4), DOS alatt COM3) a /dev/cuad2 eszközön keresztül érhetõ el tárcsázó eszközként, és a /dev/ttyd2 eszközön keresztül behívó eszközként. Mi a különbség a két eszközosztály között?

A ttydX eszközöket behívásra használjuk. Amikor tehát a /dev/ttydX eszközt blokkoló módban nyitjuk meg, akkor a hívó program egészen addig várni fog, amíg a megfelelõ cuadX eszköz inaktívvá nem válik, majd kivárja, hogy megérkezzen a hívás fogadását tolmácsoló jelzés. Amikor megnyitjuk a cuadX eszközt, gondoskodik róla, hogy a soros portot ekkor ne használja a ttydX eszköz. Ha a port szabaddá válik, egyszerûen "ellopja" a ttydX eszköztõl. Sõt, a cuadX eszközt egyáltalán nem érdekli a hívás fogadása jelzés. Ezzel a megoldással és egy automata modem segítségével a távoli felhasználók bármikor be tudnak jelentkezni a rendszerünkbe, hogy közben ugyanezzel a modemmel továbbra is tudunk tárcsázni, mivel a rendszer elintézi a többit.

15.4. Hogyan lehet engedélyezi a többportos soros vonali kártyák támogatását?

Ismét megemlítjük, hogy a rendszermag beállításával foglalkozó részben olvashatunk bõvebben a rendszermag paraméterezésének mikéntjérõl. A többportos soros vonali kártyák esetén a kártyán található mindegyik soros porthoz vegyünk fel egy-egy sio(4) bejegyzést a device.hints(5) állományába. Az IRQ és vektor értékeket azonban csak az egyiknél adjuk meg, mivel a kártyán található összes port egyetlen megszakításon fog osztozni. A következetesség kedvéért az utolsó porthoz adjuk meg a megszakítást. Ne felejtsük el még megadni a rendszermag konfigurációs állományában az alábbi opciót sem:

```
options COM_MULTIPORT
```

Az alábbi /boot/device.hints egy AST típusú négyportos soros vonali kártyát láthatunk a tizenkettedik megszakításon:

```
hint.sio.4.at="isa"
hint.sio.4.port="0x2a0"
hint.sio.4.flags="0x701"
hint.sio.5.at="isa"
hint.sio.5.port="0x2a8"
hint.sio.5.flags="0x701"
hint.sio.6.at="isa"
hint.sio.6.port="0x2b0"
hint.sio.6.flags="0x701"
hint.sio.7.at="isa"
hint.sio.7.port="0x2b8"
hint.sio.7.flags="0x701"
hint.sio.7.irq="12"
```

A `flags` paraméterrel megadott értékek azt jelzik, hogy a fõport `7` alszámmal rendelkezik (`0x700`), valamint az összes port ugyanazon a megszakításon osztozik (`0x001`).

15.5. A FreeBSD képes több többportos soros vonali kártyát ugyanazon a megszakításon keresztül használni?

Sajnos még nem. Minden kártyához másik megszakítást kell megadni.

15.6. Hogyan lehet beállítani a portok alapértelmezett paramétereit?

Ezzel kapcsolatban olvassuk el a FreeBSD kézikönyv soros kommunikációt tárgyaló részét.

15.7. Hogyan lehet a modemen betárcsázást beállítani?

Erre vonatkozóan olvassuk el a FreeBSD kézikönyv betárcsázós szolgáltatásokkal kapcsolatos részét.

15.8. Hogyan lehet buta terminálokat FreeBSD-re csatlakoztatni?

Az ezzel kapcsolatos információkat a FreeBSD kézikönyv terminálokról szóló részében találhatjuk meg.

15.9. Miért nem indul el a tip vagy cu parancs?

Elõfordulhat, hogy rendszerünkön a tip(1) és cu(1) programok csak az `uucp` felhasználón és a `dialer` csoporton keresztül tudnak hozzáférni a mûködésükhöz szükséges /var/spool/lock könyvtárhoz. A `dialer` csoport segítségével lehet szabályozni, hogy ki férhessen hozzá a modemekhez vagy a távoli rendszerekhez. Ilyenkor egyszerûen csak vegyük fel magunkat a `dialer` csoportba.

A következõ parancs kiadásával viszont ettõl függetlenül is engedélyezhetjük a rendszerünkön belül, hogy bárki használhassa a tip(1) vagy cu(1) parancsokat:

```
# chmod 4511 /usr/bin/cu
# chmod 4511 /usr/bin/tip
```

15.10. A rendszerhez csatlakozó Hayes szabványú modem támogatott - mi ilyenkor teendõ?

A FreeBSD kézikönyvben lásd ezt a választ.

15.11. Hogyan adjuk meg az AT parancsokat?

A FreeBSD kézikönyvben lásd ezt a választ.

15.12. A pn tulajdonságnál miért nem lehet @ jelet megadni?

A FreeBSD kézikönyvben lásd ezt a választ.

15.13. Hogyan lehet telefonszámokat tárcsázni parancssorból?

A FreeBSD kézikönyvben lásd ezt a választ.

15.14. Minden alkalommal meg kell adni az adatátviteli sebességet?

A FreeBSD kézikönyvben lásd ezt a választ.

15.15. Terminálszerver segítségével hogyan lehet könnyen elérni egyszerre több gépet is?

A FreeBSD kézikönyvben lásd ezt a választ.

15.16. A tip(1) képes több vonalat is használni az egyes gépek eléréséhez?

A FreeBSD kézikönyvben lásd ezt a választ.

15.17. Miért kell kétszer lenyomni a CtrlP billentyûket, hogy egyszer elküldjük ezeket?

A FreeBSD kézikönyvben lásd ezt a választ.

15.18. Miért lett hirtelen minden NAGYBETûS?

A FreeBSD kézikönyvben lásd ezt a választ.

15.19. Hogyan lehet állományokat mozgatni a tip használatával?

A FreeBSD kézikönyvben lásd ezt a választ.

15.20. Hogyan használható a zmodem protokoll a tip programmal?

A FreeBSD kézikönyvben lásd ezt a választ.

Chapter 16. Egyéb kérdések

16.1. A FreeBSD miért használ sokkal több lapozóállományt, mint a Linux®?

A FreeBSD csupán látszólag használ több helyet a lapozásra, mint a Linux®, valójában egyébként nem. A FreeBSD és a Linux® közt az egyik leglényegesebb különbség, hogy a FreeBSD valamivel elõre gondolkodik, és az összes pillanatnyilag nem használt lapot kilapozza a központi memóriából a lapozóterületre. Ezzel igyekszik minél több memóriát elõkészíteni az aktív használatra. A Linux® ezzel szemben a lapozást csak végsõ esetben használja. Ennek megfelelõen a lapozóterület gyakoribb használatát remekül ellensúlyozza a fizikai memória hatékonyabb kihasználása.

Habár a FreeBSD igyekszik ebben a tekintetben elõrelátó lenni, nem minden esetben tudja pontosan eldönteni, hogy a rendszerben mely lapokat nem használják éppen. Emiatt nem fogja az összes memóriát kilapozni, ha például egész éjszakára futni hagyjuk a gépünket.

16.2. A top miért jelez kevés szabad memóriát, miközben csak néhány program fut?

Röviden úgy válaszolhatnánk meg ezt a kérdést, hogy a szabad memória igazából elvesztegetett memória. A programok által szabadon hagyott memóriát a FreeBSD rendszermagja többnyire a lemez gyorsítótárazására használja fel. A top(1) kimenetében olvasható `Inact`, `Cache` és `Buf` értékek a lényegében különbözõ öregedési szintek szerint kategorizált tárazott adatok. A tárazás lényegében arra utal, hogy a rendszernek így nem a lassú elérésû lemezen kell a gyakran elérni kívánt adatok után kutatni, aminek köszönhetõen növekszik az összteljesítmény. A top(1) kimenetében tehát `Free` kategória alacsony értéke alapvetõen jót jelent, feltéve, ha nem *nagyon* kevés.

16.3. A chmod miért nem változtatja meg a szimbolikus linkek engedélyeit?

A szimbolikus linkekhez alapértelmezés szerint nem tartoznak engedélyek, ezért a chmod(1) ilyen esetekben az eredeti állomány engedélyeit változtatja meg. Ezért például, ha adott egy ize nevû állomány, valamint erre egy mize nevû szimbolikus link, akkor a következõ parancs mindig mûködni fog:

```
% chmod g-w mize
```

Ennek ellenére az mize engedélyei nem fognak megváltozni.

Ha egy adott könyvtárszerkezetben elhelyezkedõ állományok engedélyeit akarjuk egyszerre módosítani, akkor a `-R` opció mellett a `-H` vagy `-L` opciókat is meg kell adnunk. Errõl részletesebb információkat a chmod(1) és a symlink(7) man oldalairól tudhatunk meg.

A chmod(1) `-R` opciója *rekurzív* mûködést tesz lehetõvé. Óvatosan bánjunk a

könyvtárakkal vagy a könyvtárakra mutató szimbolikus linkekkel a chmod(1) használata során. Ha egy szimbolikus link által hivatkozott könyvtár engedélyeit akarjuk megváltoztatni, akkor a chmod(1) parancsnak ne adjunk meg semmilyen paramétert és a nevet zárjuk perjellel (/). Például, ha az ize a mize könyvtárra mutató szimbolikus link, és meg akarjuk változtatni az ize engedélyeit (ami valójában a mize engedélyeit jelenti), akkor valami ilyesmit kellene megadnunk:

```
% chmod 555 ize/
```

A név végén szereplõ perjelbõl a chmod(1) tudni fogja, hogy követnie kell a foo szimbolikus linket és így az általa hivatkozott könyvtár, a mize engedélyeit fogja megváltoztatni.

16.4. A FreeBSD képes DOS programokat futtatni?

Igen, a Portgyûjteményben található emulators/doscmd, vagyis egy DOS emulációs program segítségével.

Amennyiben a doscmd önmagában még nem lenne elegendõ, egy másik segédprogram, a emulators/pcemu segítségével emulálni tudunk egy 8088-as processzort, valamint a BIOS annyi részét, hogy futtatni tudjunk szöveges DOS alkalmazásokat. A használatához az X Window Systemre is szükségünk lesz.

Érdemes ezenkívül még megpróbálnunk a FreeBSD Portgyûjteményében található emulators/dosbox portot is. Ez az alkalmazás elsõsorban a régi DOS-os játékok futtatásához szükséges környezet emulációjára koncentrál, a helyi állományrendszerben található állományok felhasználásával.

16.5. Hogyan tudjuk az anyanyelvünkre lefordítani a FreeBSD dokumentációját?

Olvassuk el a FreeBSD Dokumentációs Projekt bevezetõjében található Fordítói GYIK-ot.

16.6. A FreeBSD.org tartományon belüli e-mail címekre küldött levelek miért pattannak vissza?

A FreeBSD.org levelezõrendszere a bejövõ levelekre vonatkozóan átvett néhány szigorúbb ellenõrzést a Postfix alkalmazástól, és ezért eldobja azokat a leveleket, amelyek formátuma hibás vagy feltehetõen szemét. A leveleink az alábbi okok miatt pattanhatnak vissza:

- A levelet olyan név- vagy IP-tartományból küldtük, ahonnan korábban levélszemetet küldtek, ezért feketelistára került.

  A FreeBSD levelezõ szerverei eldobnak minden olyan levelet, amelyek feketelistás tartományokból érkeznek. Ha olyan cégen vagy tartományon keresztül akarunk küldeni,

amelyik levélszemetet gyárt vagy továbbít, akkor váltsunk szolgáltatót.

- A levél törzse csak HTML kódot tartalmaz.

  A leveleinket egyszerû szöveges formátumban küldjük. Állítsuk be a levelezõ kliensünket erre.

- A `FreeBSD.org` címen üzemelõ levelezõ szerver nem tudta a csatlakozó gép IP-címét szimbolikus névre feloldani.

  Az ellenkezõ irányú névfeloldás sikeressége alapvetõ követelmény a levelek fogadásához. Gondoskodjunk róla, hogy a levelezõ szerverünk IP-címével mûködjön az inverz névfeloldás, Sok otthoni szolgáltatás (DSL, kábel, betárcsázós stb. kapcsolat) erre nem ad lehetõséget. Ilyenkor a leveleinket próbáljuk meg a szolgáltatónk levelezõ szerverein keresztül küldeni.

- Az SMTP protokoll EHLO/HELO részében megadott hálózati név nem oldható fel valós IP-címre.

  Egy teljes, feloldható hálózati név elegendhetetlen a levél elfogadásához szükséges SMTP párbeszéd érvényességéhez. Ha nincs hivatalosan bejegyzett hálózati nevünk, akkor a szolgáltató levelezõ szervereit kell használnunk a levél elküldéséhez.

- A küldött üzenet azonosítója (Message ID) végén a `localhost` szerepel.

  Egyes levelezõ kliensek rossz azonosítónak hoznak létre az üzenetekhez, ezért a rendszer nem hajlandó elfogadni ezeket. Ilyenkor vagy rávesszük valahogy a levelezõ kliensünket, hogy rendes azonosítókat készítsen, vagy úgy állítjuk be a levéltovábbítónkat, hogy érvényes azonosítókra írja át.

16.7. Hogyan lehet egyszerûen FreeBSD rendszereket elérni?

Habár a FreeBSD maga nem nyújt akárki számára hozzáférést a saját szervereihez, mások viszont kínálnak bárki által elérhetõ UNIX® rendszereket. Ennek költsége és minõsége szolgáltatónként változik.

Az Arbornet, Inc, vagy másik nevén *M-Net* 1983 óta szolgáltat nyílt hozzáférést UNIX® típusú rendszerekhez. Egy System III alapokon mûködõ Altos rendszerrõl a 1991-ben BSD/OS-re váltottak, majd 2000 júliusában aztán FreeBSD-re váltottak. Az *M-Net* telnet és SSH szolgáltatásokon keresztül is elérhetõ, és lényegében a FreeBSD alatt elérhetõ összes programhoz enged egy alapvetõ hozzáférést. A hálózati hozzáférés azonban csak a tagok és a támogatók számára engedélyezett. Ez egy non-profit szervezet. Az *M-Net* rendelkezik üzenõfallal (bulletin board system, BBS) és interaktív csevegõrendszerrel is.

A Grex az *M-Net* szolgáltatásához hasonlóan ugyanúgy kínál üzenõfalat és csevegési lehetõséget. Többségében azonban Sun™ 4M gépeik vannak, amelyen SunOS™ fut.

16.8. Mi az a sup és hogyan lehet használni?

A SUP mozaikszó mögött a "Software Update Protocol" ("Szoftverfrissítési protokoll") áll, amelyet fejlesztési fák szinkronban tartására dolgoztak ki a Carnegie-Mellon Egyetemen. Régebben ennek

segítségével tartották frissítették magukat a fejlesztõi források különbözõ tükrözései a FreeBSD Projekten belül.

A SUP nem kifejezetten egy sávszélesség-takarékos megoldás, és egy ideje már nyugdíjba vonult. A forrásainkat jelen pillanatban a CVSup használatával tudjuk frissíteni.

16.9. Hogy hívják azt a cuki kis vörös fickót?

Igazából nincs neve, mindenki egyszerûen csak "BSD démonnak" nevezi. Ha mégis hívni szeretnénk valahogy, akkor szólítsuk csak "beastie"-nek, ugyanis a "beastie" kiejtése megegyezik a "BSD" szóéval ("bíeszdi").

A BSD démonról a saját honlapján tudhatunk meg többet.

16.10. Felhasználható a BSD démon képe?

Talán. A BSD démon jogait Marshall Kirk McKusick birtokolja. A felhasználás pontos lehetõségeivel kapcsolatban olvassuk el Statement on the Use of the BSD Daemon Figure címû írást.

Röviden úgy foglalhatnánk össze, hogy ízléses stílusban a saját céljainkra mindaddig nyugodtan felhasználhatjuk a képet, amíg megemlítjük az eredeti szerzõt. Ha kereskedelmi céljaink vannak, akkor írjunk Kirk McKusick <mckusick@FreeBSD.org> címére. A pontosabb részleteket a BSD démon honlapján olvashatjuk.

16.11. Található valahol felhasználható kép a BSD démonról?

EPS és XFig formátumú rajzok a /usr/shared/examples/BSD_daemon/ könyvtárban vannak.

16.12. A levelezési listákon szerepeltek ismeretlen kifejezések vagy rövidítések. Hol lehet ezeknek utánanézni?

Olvassuk el a FreeBSD szakkifejezéseinek gyûjteményét.

16.13. Miért fontos annyira a biciklitároló színe?

Erre röviden úgy adhatnánk választ, hogy ezzel igazából nem kell annyira törõdnünk. Ha viszont valamivel terjedelmesebben akarunk válaszolni, akkor azt mondhatnánk, hogy azért, mert egy biciklitároló megépítése még nem tántorít el senkit sem a választott szín kritizálásától és az átfestésének fontolgatásától. Ez a metafora alapvetõen arról szól, hogy nem kell feltétlenül minden apró részletrõl vitatkoznunk csupán azért, mert jobban értünk hozzá. Sokak tapasztalata szerint ugyanis a változtatásokhoz kapcsolódó megjegyzések által gerjesztett zaj fordítottan arányos az adott változtatás bonyolultságával.

A még hosszabb és teljesebb válasz eredetileg egy nagyon hosszú és fárasztó vita eredményeképpen keletkezett, amikor arról esett szó, hogy a sleep(1) törtekkel dolgozzon-e vagy sem. Erre válaszul küldte Poul-Henning Kamp <phk@FreeBSD.org> az azóta híressé vált "A bike shed (any color will do) on greener grass..." ("(Bármilyen színû) biciklitároló megfelelne egy zöldebb gyepen...") címû levelét. Ebbõl szeretnénk most idézni:

Poul-Henning Kamp <phk@FreeBSD.org>, freebsd-hackers, 1999. október 2. "Mirõl is szól ez a biciklitároló?" - kérdezték tõlem sokan.

Ez egy hosszú, vagy még inkább régi történet, amely azonban valójában meglehetõsen rövid. C. Northcote Parkinson "Parkinson törvénye" címmel írt egy könyvet az 1960-as évek elején, amelyben elég nagy betekintést adott a vezetés dinamikájába.

[a könyv részletes bemutatását most kihagyjuk]

A konkrét példában egy biciklitároló szerepel egy atomerõmûvel szemben, szóval ez is eléggé jól érzékelteti a könyv korát.

Parkinson ezen keresztül bemutatja, hogyan kell egy igazgatói tanács elé járulni egy több millió vagy akár milliárd dolláros atomerõmû megépítéséhez, azonban egy egyszerû biciklitároló megépítésekor könnyen véget nem érõ vitatkozásba bonyolódhatunk.

Parkinson elmagyarázza, mindez azért van, mert egy atomerõmû annyira óriási, drága és bonyolult, hogy az emberek egyszerûen nem értik meg. Ezért nem szólnak semmit és megnyugtatják magukat a feltételezéssel, hogy valaki más korábban már biztosan utánajárt a részleteknek. Richard P. Feynmann is könyveiben rengeteg érdekes és nagyon találó példát ad ezekre Los Alamossal kapcsolatban.

Vegyünk ezzel szemben most egy biciklitárolót. Bárki képes egy hétvége alatt összetákolni egy ilyet és még így is marad ideje megnézni a meccset. Ezért nem számít, mennyire jól megfogalmazott, elõkészített és logikus is a javaslatunk, valaki biztosan meg fogja ragadni a lehetõséget, hogy az orrunk elõtt fitogtassa a képességeit és megmutassa magát: õ bizony *itt járt*.

Dániában ezt mi úgy hívjuk, hogy "otthagyjuk a kezünk nyomát". Ez mindössze a személyes büszkeségrõl és tekintélyrõl szól, vagyis hogy végre elmondhassuk: "Ezt nézd! *Én* csináltam." Ez ugyan leginkább a politikusokra jellemzõ, de alapvetõen minden emberben ott él. Gondoljunk csak a friss betonban hagyott lábnyomokra.

Chapter 17. Mókás dolgok a FreeBSD-vel kapcsolatban

17.1. Mennyire hûsít a FreeBSD?

Kérdés: Mérte már valaki, hogy a FreeBSD futása közben mennyire melengeti meg a számítógépet? Úgy hírlik, a Linux® ebben a tekintetben sokkal jobb, mint a DOS, de FreeBSD-rõl még nem ismert ezzel kapcsolatban semmi. Mondjuk, elég tüzesnek tûnik.

Válasz: Nem, de korábban már számos tesztet végeztünk bekötött szemû önkénteseken, akiknek elõzetesen 250 mikrogram LSD-25-öt adagoltak. A tesztalanyok 35 százaléka szerint a FreeBSD kissé narancsos ízû volt, míg a Linux® inkább a rózsaszín ködhöz hasonlított. A hõmérséklettel kapcsolatban azonban egyik csoport sem észlelt komolyabb változást. Végül aztán teljesen el kellett vetnünk a kísérlet eredményeit, mert menet közben túlságosan sok önkéntes kóborolt el, és ezzel torzították a mérések eredményeit. A legtöbb önkéntes azóta is Apple-nél van, és azóta is egy új "színes, szagos" grafikus felületen dolgoznak. Szép kis felfordulás!

Komolyan: a FreeBSD és a Linux® is egyaránt a processzorokban található HLT (halt) utasítást használja arra, hogy az üresjáratban levõ rendszer energiafogyasztását és ezáltal hõtermelését is valamennyire mérsékelje. Emellett még az APM (Advanced Power Management) is támogatott, így a FreeBSD akár tetszés szerint alacsonyabb energiafogyasztású módba is tudja tenni a processzort.

17.2. Mi mocorog a memóriamodulokban?

Kérdés: A FreeBSD csinál valami "szokatlan" a rendszermag fordítása közben, ami miatt a memóriák felõl mocorgást lehet hallani? Amikor fordítok (vagy egy rövid ideig, amikor az indításkor a rendszer keresi a floppymeghajtót) valamilyen furcsa mocorgásszerû hang jön a memóriamodulokból.

Válasz: Igen! Gyakran utalnak a BSD rendszerek dokumentációiban mindenféle "démonokra", és ezzel kapcsolatban a legtöbb ember nem is tudja, hogy ezek valójában apró, öntudatos, fizikailag nem létezõ lények, amelyek a rendszer indulása után megszállják a számítógépünket. A memóriából kiszûrõdõ mocorgás hangja igazából a démonok közti magas frekvenciás beszélgetésbõl ered, amikor éppen arról egyeztetnek, hogy miként birkózzanak meg a különbözõ rendszeradminisztrációs feladatokkal.

Ha teljesen megõrjít minket ez a zajongás, akkor úgy tudunk tõlük megszabadulni, ha kiadjuk DOS-ból a jó öreg `fdisk /mbr` parancsot. Ekkor viszont ne lepõdjünk meg, ha netalán visszalõnének és próbálnak minket megállítani. Ha eközben a hangszóróinkból Bill Gates sátáni kacaja harsanna fel, akkor rohanjunk és ne is nézzünk többet vissza! A BSD démonok támogatásától mentesen a Windows® és a DOS ikerördögei ilyenkor gyakran visszaszerzik gépünk felett a teljes irányítást és ezzel örök szenvedésre kárhoztatják gyarló lelkünket. Ennek tudatában lehet, hogy mégis csak jobb lenne, ha egyszerûen csak hozzászoknánk azokhoz a furcsa hangokhoz, nem?

17.3. Hány FreeBSD fejlesztõ kell egy villanykörte kicseréléséhez?

Ezeregyszázhatvankilenc:

Huszonhárman panaszkodnak a -current listán, hogy már megint kiment a villany.

Négyen erre azt válaszolják, hogy ez csak konfigurációs probléma, ezért ennek a -questions listán a helye.

Hárman írnak róla hibajelentést, de ezek közül az egyik ráadásul tévesen a `doc` kategóriába kerül, és csak annyi áll benne, hogy "sötét van".

Erre az egyikük beszerel egy kipróbálatlan villanykörtét, amitõl nem mûködik a rendszer többi része, így öt perc múlva ki is szereli.

Nyolcan leszidják a hibajelentések íróit, hogy nem mellékelték a javítást a jelentéseik mellé.

Öten siránkoznak, hogy nem mûködik a rendszer.

Harmincegyen erre azt válaszolják, hogy nekik minden remekül mûködik, és az érintettek minden bizonnyal pont rosszkor frissítettek.

Egy küld egy új villanykörtét a -hackers listára.

Erre egy rászól, hogy õ már három évvel ezelõtt megcsinálta ugyanezt, de amikor beküldte a -current listára, akkor senki sem foglalkozott vele, és egyébként sem szereti a hibajelentéseket. Emellett ráadásul az új villanykörte egyébként sem tetszik.

Huszonheten nekiállnak skandálni, hogy a villanykörték nem tartoznak az alaprendszerbe, ezért a committerek a közösség megkérdezése nélkül nem csinálhatnak semmit, és különben is: *Mi errõl a -core véleménye?*

Kétszázan eközben megvitatják, milyen színû legyen a biciklitároló.

Hárman jelzik, hogy a javítás nem felel meg a style(9) elõírásainak.

Tizenheten megjegyzik, hogy az újonnan javasolt villanykörte GPL licenccel rendelkezik.

Ötszázhatvankilencen valóságos vitaözönt indítanak a GPL, a BSD, MIT és NPL licencek elõnyeit illetõen, majd megjegyzéseket tesznek különféle meg nem nevezett FSF alapítók személyes higéniajára.

Heten a vita bizonyos részeit átviszik a -chat és -advocacy listákra.

Egy végül beszereli a javasolt villanykörtét, de az valamivel mintha halványabban világítani, mint az elõzõ.

Ketten leszólják a szerelést, és összekapnak azon, hogy most akkor a FreeBSD inkább maradjon sötétségben vagy érje be a halványabb világítással.

Negyvenhárman rikácsolva követelik a halványan világító villanykörte kiszerelését és panaszukat megírják a -core listára.

Tizenegyen egy kisebb villanykörtét kérnek, mert ha majd portolni akárják a Tamagotchijukra a rendszert, akkor ott is használható legyen.

Hetvenhárman felemelik a szavukat a -hackers és -chat listákon felerõsödött zaj miatt, és tiltakozásul leiratkoznak ezekrõl a listákról.

Tizenhárman erre egy "leiratkozom", " Hogyan kell innen leiratkozni?" vagy "Kérlek, vegyetek le errõl a listáról" témájú levelet küldenek a megszokott stílusban.

Egy eközben beszerel végre egy mûködõ villanykörtét, miközben mindenki azzal van elfoglalva, hogy szidja a másikat, így szinte észre sem veszik.

Harmincegy ezután hozzáteszi, hogy az új villanykörte 0,364 százalékkal jobban világítana, ha TenDRA-val csinálták volna (akkor viszont kocka alakú lenne) és a FreeBSD-nek ezért a GCC helyett TenDRA-t kellene használnia.

Egy valaki megemlíti, hogy az új villanykörtén nincs is burkolat.

Kilencen (beleértve a hibajelentések íróit) azt kérdezgetik folyton, hogy "Mi az az MFC?".

Ötvenheten két hét múlva kezdenek el panaszkodni, hogy a villanykörte kiment.

Nik Clayton <nik@FreeBSD.org> hozzáteszi:

Nagyon jót nevettem ezen.

Közben az jutott az eszembe, hogy "Várjunk csak, nem kellene valahol a felsorolásban lennie egy "egy, aki pedig ledokumentálja" résznek?"

És akkor végre megértettem :-)

Thomas Abthorpe <tabthorpe@FreeBSD.org> szerint: "Egy sem, mert a *valódi* FreeBSD fejlesztõk nem félnek a sötétben!"

17.4. Hova kerül a /dev/null eszközre küldött adat?

A processzoron található speciális adatsüllyesztõbe kerül, majd hõvé alakul és elszállítja a felszerelt hûtõborda és ventillátor. Ezért is annyira fontos a processzor hûtése: az emberek minél gyorsabb géppel rendelkeznek, annál inkább gondatlanná válnak és annál több adat köt ki a /dev/null eszközben. Ha sikerül letörölnünk a /dev/null eszközt (amivel így lényegében letiltjuk a processzor adatsüllyesztõjét), akkor a processzorunk ugyan kevésbé fog melegedni, viszont gyorsan eldugul a sok adattól és furcsán kezd el viselkedni. Ha nagyon gyors hálózati kapcsolattal rendelkezünk, akkor úgy is le tudjuk hûteni a processzorunkat, ha folyamatosan olvassuk a /dev/random eszközt és valahova elküldjük az eredményt. Ekkor viszont vigyázzunk arra, hogy ezzel a módszerrel könnyen túlmelegedhet a hálózati kártyánk és a gyökér állományrendszerünk, valamint a szolgáltató sem fog örülni ennek, mert akkor a felesleges hõ náluk keletkezik. Általában viszont jó a hûtésük, ezért ha okosan csináljuk, akkor semmi gondunk nem származik belõle.

Paul Robinson hozzáteszi:

Vannak még más módszerek is. Minden jó rendszergazda tudja, hogy szokás a képernyõre is folyamatosan adatot küldeni, mert így a pixik is vidámabbak lesznek. A képernyõt formázó pixik (melyek gyakran tévesen és hibásan "pixeleknek" hívnak) a fejükön viselt kalapok szerint három csoportba sorolhatóak (vörös, zöld vagy kék), és annak megfelelõen bújnak elõ (illetve mutatják meg a kalapjukat), hogy kapnak-e enni. A videokártyák felelõsek azért, hogy a kapott adatokból pixiétel készüljön és hogy az eljusson a pixikhez - minél drágább a kártya, annál jobb minõségû az elõállított étel, és annál fegyelmezettebben viselkednek a pixik. Állandó cirogatásra is szükségük van - ez a képernyõvédõk feladata.

Az elõbbi javaslatot azzal tudnám még kiegészíteni, hogy a /dev/random eszköztõl származó adatokat akár a konzolra is küldhetjük, így a pixiket is jól tudjuk lakatni. Ezzel együtt nem jár semmilyen hõtermelés, viszont a pixik boldogok lesznek és így könnyen meg tudunk szabadulni a felesleges adatoktól is, még úgy is, ha kissé zavarosnak tûnik közben a kép.

Mellesleg mint az egyik nagy szolgáltató egykori rendszergazdája elmondhatom, hogy mivel tapasztalatom szerint a szerverszobában nehéz tartani a megfelelõ hõmérsékletet, ezért nem ajánlom senkinek a felesleges adatok átküldését a hálózaton. A csomagok közvetítésével és irányításával foglalkozó tündérek sem különösebben szoktak örülni ennek.

Chapter 18. Témák haladóknak

18.1. Honnan lehet többet megtudni a FreeBSD belsõ felépítésérõl?

Jelen pillanatban csak egyetlen mû foglalkozik az operációs rendszerek felépítésével a FreeBSD szemszögébõl, név szerint a Marshall Kirk McKusick és George V. Neville-Neil által írt "The Design and Implementation of the FreeBSD Operating System" címû könyv (ISBN 0-201-70245-2), amely a FreeBSD 5.*X* változatára koncentrál.

Emellett a UNIX® típusú rendszerek használatával kapcsolatos ismeret remekül alkalmazható a FreeBSD esetén is.

A témához tartozó többi könyvet a kézikönyv Az operációs rendszerek belsõ mûködésével foglalkozó irodalomjegyzékben találhatjuk meg.

18.2. Hogyan lehet bekapcsolódni a FreeBSD fejlesztésébe?

Pontosabb tanácsokat akkor kapunk, ha elolvassuk a FreeBSD fejlesztésérõl szóló cikket. Nagyon is számítunk mindenki segítségére!

18.3. Mik azok a pillanatkiadások és kiadások?

Jelenleg három aktív és félig aktív ág van a FreeBSD CVS repositoryjában. (A korábbi ágakat már csak nagyon ritkán módosítják, ezért is csak három aktív fejlesztési ágon fejlesztenek):

- `RELENG_7` avagy *7-STABLE*
- `RELENG_8` avagy *8-STABLE*
- `HEAD` avagy *-CURRENT* avagy *9-CURRENT*

A `HEAD` nem olyan ág, mint a másik kettõ. Ez egyszerûen csak "*a jelenlegi, még el nem ágaztatott fejlesztési irány*" jelentéssel bír, amire pedig sokszor röviden csak *-CURRENT* néven hivatkoznak.

Jelen pillanatban a *-CURRENT* a 9.*X* fejlesztési irányát képviseli; az `6-STABLE` ág, a RELENG_6, 2005 novemberében, a `7-STABLE` ág, a RELENG_7, 2008 februárjában, míg a `8-STABLE` ág, a RELENG_8, 2009 novemberében vált le a "-CURRENT" ágból.

18.4. Hogyan lehet saját kiadást készíteni?

Olvassuk el a kiadások készítésérõl szóló cikket.

18.5. A make world parancs miért írja felül a korábban telepített binárisokat?

Mert alapvetõen ez lenne a cél: ahogy a neve is sugallja, a rendszer újrafordítása, vagyis a `make world` parancs feladata a rendszerben található összes bináris újrafordítása, aminek eredményeképpen egy tiszta és összefüggõ környezetet kapunk (ezért is tart ilyen sokáig).

Ha a `make world` vagy a `make install` parancs futtatása elõtt megadjuk a `DESTDIR` környezeti változót, akkor a frissen létrehozott binárisok az általa mutatott könyvtárba fognak kerülni pontosan úgy, ahogy az eredeti rendszer. Az osztott könyvtárak bizonyos módosításai és egyes programok fordítása azonban könnyen térdre kényszerítheti a `make world` futását.

18.6. Miért nem forgó (round robin) névfeloldással lehet elérni a CVSup szervereket és így megosztani köztük a terhelést?

Habár a CVSup tükrözések óránként frissítik magukat a központi CVSup szerverrõl, maga a frissítés azonban bármikor megtörténhet. Ennek következményeképpen egyes szervereken frissebb kód található, miközben a többin még az egy órával ezelõtti állapot szerepel. Ha a `cvsup.FreeBSD.org` forgó névfeloldással mûködne, akkor a felhasználók mindig egy véletlenszerûen választott CVSup szervert kapnának, és ezért a CVSup egymás utáni futtatásakor könnyen elõfordulhatna, hogy a rendszer régebbi forrásait kapjuk vissza.

18.7. A -CURRENT forrásait korlátozott interneteléréssel is lehet követni?

Igen, ezt a CTM használatával *anélkül* is megtudjuk tenni, hogy le kellene töltenünk az egész forrásfát.

18.8. Hogyan lehet 1392 KB-os darabokra felosztani az egyes terjesztéseket?

Az újabb BSD alapú rendszerekben a split(1) parancsnak már van egy `-b` paramétere, amellyel tetszõleges méretûre fel tudunk darabolni állományokat.

Íme erre egy példa a /usr/src/release/Makefile állományból:

```
ZIPNSPLIT=              gzip --no-name -9 -c | split -b 1392k -
```

18.9. Hova lehet küldeni a rendszermaghoz írt kiegészítéseket?

Erre vonatkozóan vessünk egy pillantást a FreeBSD továbbfejlesztésérõl szóló cikkre.

Köszönjük, hogy gondolt ránk!

18.10. A rendszer hogyan érzékeli és inicializálja a Plug and Play ISA kártyákat?

Frank Durda IV (uhclem@nemesis.lonestar.org) válasza:

Dióhéjban úgy tudnám ezt elmagyarázni, hogy van néhány I/O port, amelyet lekérdezve a PnP kártya képes válaszolni, hogy elérhetõ-e. Ezért a PnP eszközök keresése azzal kezdõdik, hogy a rendszer felteszi a kérdést, van-e PnP kártya a számítógépben. Erre aztán a különbözõ kártyák a típusuk megjelölésével válaszolnak, amelyet ugyanezen az I/O porton kell visszaolvasni, így ha már legalább egy bitet beállít valaki, akkor folytatható a keresés. Ezután a keresést végzõ kódrész letiltja az `X` alatti (a Microsoft® és az Intel® által kiosztott) azonosítóval rendelkezõ kártyákat, majd ismét megnézi, hogy valaki továbbra is válaszol-e. Amennyiben a válasz `0`, az arra utal, hogy már nincs aktív kártya az `X` azonosító felett. Ezt követõen a rendszer megpróbálkozik az `X` alatti azonosítók lekérdezésével. Végül folytatja az `X` alatti keresést az `X -(korlát / 4)` feletti azonosítók letiltásával, majd megismétli az iménti kérdést. Ezzel a félig-meddig bináris keresési módszerrel aztán képes 2 lépésnél jóval kevesebbõl felderíteni a rendszerünkben megtalálható PnP kártyákat.

Az azonosítók két 32 bit hosszúságú mezõbõl (ezért írtunk az elõbb 2 lépést) és egy 8 bites ellenõrzõösszegbõl állnak. Az elsõ 32 bit a gyártót azonosítja. Ugyan soha nem vallják be, de úgy tûnik, hogy még ugyanannak a gyártónak is lehetnek eltérõ gyártóazonosítóval rendelkezõ kártyái. A gyártók számára fenntartott 32 bites mezõ ezért valamennyire túlzás.

A második 32 bit lehet a kártya sorozatszáma vagy bárki más, amely alapján egyértelmûen beazonosítható. A gyártó ugyanazzal a 32 bites értékkel nem gyárthat egy másik kártyát, csak abban az esetben, ha a másik 32 bit is eltér. Ennek köszönhetõen egy gépen belül még az azonos típusú kártyák is el fognak térni 64 biten.

Az iménti 32 bites csoportok nem lehetnek teljesen nullák, ezért lehetséges, hogy a bináris keresés során a válaszban legalább egy bit mindig aktív lesz.

Miután a rendszer sikeresen beazonosította a rendelkezésre álló kártyákat, egyenként újra elindítja ezeket (ugyanazon az I/O porton keresztül), és megpróbálja kitalálni, hogy az adott eszközöknek milyen erõforrásokra van szüksége, milyen megszakítást akarnak használni stb. Az összes kártyától lekérdezi ezeket az információkat.

Az így megszerzett információkat aztán még kiegészíti a merevlemezen vagy az MLB BIOS-ban található ECU állományok tartalmával. Az ECU és az MLB BIOS PnP támogatása általában viszont nem valódi, és az ilyen eszközök igazából nem is állítanak be semmit maguktól. A BIOS és az ECU átvizsgálása azonban segít a felderítést végzõ rutinnak értesíteni a tényleges PnP eszközöket, hogy ne foglaljanak el olyan erõforrásokat, amelyeket a rendszer nem tud áthelyezni.

Ezután a PnP eszközöket a kód még egyszer végigjárja és átadja nekik a mûködésükhöz szükséges I/O, DMA, IRQ és memóracímek hozzárendeléseit. Az eszközök ekkor a megadott helyeken elérhetõvé válnak és úgy is maradnak a rendszer következõ indításáig, de igazából semmi sem rögzíti ezeket.

Talán túlságosan is egyszerûsítettem a fentieket, de szerintem már ennyi is elegendõ az alapok megértéséhez.

A Microsoft® néhány elsõdleges nyomtatási állapotot jelzõ portot átrakott PnP-re, azzal a címszóval, hogy egyik kártya sem kódolta át ezeket a címeket az ellenkezõ I/O ciklusok számára. Találtam is egy eredeti IBM nyomtatókártyát, amely valóban át tudta írni az állapotjelzõ portot a PnP kezdeti változataiban, de arra a Microsoft® csak annyit mondott, hogy "fogós". Ezért a nyomtatási állapotot jelzõ portot a címek beállítására használja, illetve még a `0x800`-as portot és egy harmadik I/O portot valahol a `0x200` és a `0x3ff` környékén.

18.11. Hogyan lehet fõeszközazonosítót rendelni egy általunk fejlesztett meghajtóhoz?

2003 februárja óta a FreeBSD képes dinamikusan és önmûködõen futás közben lefoglalni fõeszközazonosítókat a meghajtóknak (lásd devfs(5)), ezért erre tulajdonképpen már nincs szükség.

18.12. A könyvtárakra vonatkozóan milyen más kiosztási házirendek léteznek még?

A könyvtárak más fajta kiosztására vonatkozóan annyit tudok válaszolni, hogy a jelenleg is alkalmazott sémát az 1983-ban megalkotott változata óta változatlanul használjuk. Eredetileg a gyors állományrendszerhez készítettem, de soha nem ragaszkodtam hozzá. Remekül megoldja a cilindercsoportok betelésének problémáját, azonban sokan megjegyezték már, hogy a find(1) esetén gyengén mûködik. A legtöbb állományrendszert mélységi bejárással hozzák létre, így a könyvtárak szétszóródnak a cilindercsoportok közt és ezzel a késõbbi mélységi keresések számára a lehetõ legrosszabb helyzetet alakítják ki. Ha valaki például tudja elõre a létrehozni kívánt könyvtárak számát, akkor ezt úgy lehet megoldani, ha a mûvelet során `(összes / cilindercsoportok)` mennyiségû könyvtárat hozunk létre az egyes cilindercsoportokban. Ennek meghatározására nyilvánvalóan lehet adni valamilyen heurisztikát. Már egy kisebb elõre rögzített szám, mint például a 10 kiválasztása is legalább egy nagyságrendnyi javulást jelent. Ha szeretnénk megkülönböztetni az állományrendszerek visszaállítását a hagyományos mûködéstõl (amire a jelenlegi algoritmus sokkal érzékenyebb), akkor érdemes tizes csoportokba összefogni a könyvtárakat, feltéve, hogy 10 másodpercen belül hoztuk létre ezeket. Mindenesetre elmondható, hogy ezzel nyugodtan lehet kísérletezni.

Kirk McKusick <mckusick@FreeBSD.org>, 1998 szeptembere

18.13. Hogyan lehet kinyerni a legtöbb információt a rendszermag összeomlásából?

Általában így néz ki a rendszermag összeomlása:

```
Fatal trap 12: page fault while in kernel mode
fault virtual address   = 0x40
fault code              = supervisor read, page not present
instruction pointer     = 0x8:0xf014a7e5
stack pointer           = 0x10:0xf4ed6f24
frame pointer           = 0x10:0xf4ed6f28
code segment            = base 0x0, limit 0xfffff, type 0x1b
                        = DPL 0, pres 1, def32 1, gran 1
processor eflags        = interrupt enabled, resume, IOPL = 0
current process         = 80 (mount)
interrupt mask          =
trap number             = 12
panic: page fault
```

Amikor egy ilyen üzenetet látunk, akkor nem elegendõ újra elõcsalni a hibát és beküldeni. Az utasításszámláló ("instruction pointer") értéke ugyan nagyon fontos, de sajnos konfigurációk szerint eltérhet. Más szóval úgy fogalmazhatnék, hogy ennek az értéke a használatban levõ rendszermag értékétõl függõen változhat. Ha a GENERIC rendszermagot használjuk valamelyik kiadásból, akkor viszont már elképzelhetõ, hogy valaki más is le tudja nyomozni a hibát okozó függvényt. Ha viszont egy saját beállításokkal rendelkezõ rendszermagot használunk, akkor egyedül csak *mi* vagyunk képesek megmondani a hiba pontos helyét.

Ezért a javaslatom a következõ:

1. Jegyezzük le az utasításszámláló értékét. A `0x8:` rész ebben az esetben annyira nem fontos, egyedül csak a `0xf0xxxxxx` részre van szükségünk.
2. A rendszer újraindításakor írjuk be a következõt:

   ```
   % nm -n /a.hibát.okozó.rendszermag | grep f0xxxxxx
   ```

   ahol az `f0xxxxxx` az utasításszámláló értéke. Könnyen elõfordulhat, hogy ilyenkor még nem találunk egyezést, mivel a rendszermag szimbólumtáblájában csak az egyes függvények belépési pontjai találhatóak, és ha az utasításszámláló általában valamelyikük belsejébe mutat, nem az elejükre. Ha tehát nem még látunk semmit, akkor egyszerûen hagyjuk el az utolsó számjegyet és próbálkozzunk így:

   ```
   % nm -n /a.hibát.okozó.rendszermag | grep f0xxxxx
   ```

   Ha még ez sem hoz eredményt, akkor vágjunk le a végérõl egy újabb számjegyet. Egészen addig csináljuk, amíg nem kapunk valami értékelhetõ eredményt. Ilyennek tekintjük például azokat a függvényeket, amelyek a hibát okozhatták. Ez ugyan egy nem annyira pontos felderítési eszköz, viszont még ez is jobb a semminél.

A legjobb viszont mégis az, amikor sikerül lementeni a hiba bekövetkezésekor a memória tartalmát,

majd a kgdb(1) használatával elõbányászni belõle egy hívási láncot.

Ehhez többnyire a következõ módszer javasolt:

1. A rendszermag konfigurációs állományába (/usr/src/sys/arch/conf/RENDSZERMAGKONFIG) vegyük fel a következõ sort:

```
makeoptions     DEBUG=-g          # A rendszermag fordítása gdb(1)
szimbólumokkal
```

2. Lépjünk be a /usr/src könyvtárba:

```
# cd /usr/src
```

3. Fordítsuk le a rendszermagot:

```
# make buildkernel KERNCONF=RENDSZERMAGKONFIG
```

4. Várjuk meg, amíg a make(1) befejezi a fordítást.

```
# make installkernel KERNCONF=RENDSZERMAGKONFIG
```

5. Indítsuk újra a gépet.

A KERNCONF használata nélkül a GENERIC rendszermag fordul és telepítõdik.

A make(1) programnak a folyamat végeredményeként két rendszermagot kell készítenie: a /usr/obj/usr/src/sys/RENDSZERMAGKONFIG/kernel és a /usr/obj/usr/src/sys/RENDSZERMAGKONFIG/kernel.debug. Ezek közül a kernel /boot/kernel/kernel néven mentõdik el, miközben a kernel.debug használható nyomonkövetésre a kgdb(1) programmal.

A rendszer csak akkor fogja elmenteni összeomláskor a memória tartalmát, ha az /etc/rc.conf állományban beállítjuk a dumpdev értékét a lapozóállományt tároló partícióra (vagy az AUTO értékre). Ennek hatására az rc(8) szkriptek a dumpon(8) paranccsal képesek engedélyezni a memória lementését. A dumpon(8) természetesen manuálisan is elindítható. Az összeomlást követõen a memória lementett tartalmához a savecore(8) programmal férhetünk hozzá. Amikor viszont az /etc/rc.conf állományban megadjuk a dumpdev értékét, az rc(8) szkriptek maguktól lefuttatják a savecore(8) parancsot és átrakják a mentést a /var/crash könyvtárba.

A FreeBSD által létrehozott memóriamentések mérete általában a számítógépünkben levõ fizikai memória mennyiségével egyezik meg. Tehát ha 512 MB RAM van a gépünkben, akkor egy 512 MB méretû mentést fogunk kapni. Ezért gondoskodjunk róla, hogy a /var/crash könyvtárban mindig legyen elegendõ hely

az állomány tárolásához. A savecore(8) kézzel is lefuttatható, és ilyenkor a memóriát akár egy másik könyvtárba is menthetjük. A mentés méretét `options MAXMEM=N` beállítással is korlátozhatjuk, ahol az *N* értéke a rendszermag által használható memória mérete KB-okban. Például, ha 1 GB RAM van a gépünkben, de a rendszermag által használható memóriát lekorlátozzuk 128 MB-ra, akkor a mentés mérete sem 1 GB lesz, hanem csak 128 MB.

Ahogy sikerült hozzájutnunk a memóriamentéshez, azonnal is kérhetünk a kgdb(1) használatával egy hívási láncot belõle:

```
% kgdb /usr/obj/usr/sys/RENDSZERMAGKONFIG/kernel.debug /var/crash/vmcore.0
(kgdb) backtrace
```

Elõfordulhat, hogy ilyenkor több oldalnyi információ özönlik hirtelen a képernyõre, ezért javasolt ezeket lementeni a script(1) programmal. A nyomkövetési szimbólumokat is tartalmazó rendszermag esetén még akár azt a sort is megkapjuk a rendszermagon belül, ahol a hiba történt. A hívási láncot általában alulról felfelé kell olvasni, és ebbõl deríthetõ, hogy pontosan milyen események is vezettek az összeomláshoz. A kgdb(1) használatával még a különbözõ változók és struktúrák értékeit is meg tudjuk vizsgálni, így még többet megtudhatunk a rendszer állapotáról az összeomlás pillanatában.

Ha az iméntiek mentén nagyon fellelkesültünk volna és van egy másik számítógépünk is, akkor a kgdb(1) akár távoli nyomkövetésre is beállítható, aminek köszönhetõen a kgdb(1) használatával az egyik rendszeren meg tudjuk állítani a másikon futó rendszermagot, ellenõrizhetjük a viselkedését, akárcsak bármelyik más felhasználói program esetében.

Ha netalán engedélyeztük volna a `DDB` beállítást, és a rendszermag beleáll a nyomkövetõbe, akkor a rendszert mi magunk is össze tudjuk omlasztani (és így a memóriát elmenteni) a `ddb` parancssorában a `panic` parancs kiadásával. Ilyenkor a nyomkövetõ általában még egyszer megáll az összeomláskor. Ekkor a `continue` paranccsal fejeztethetjük be a memória lementését.

18.14. A dlsym() függvény miért nem mûködik már az ELF állományokra?

Az ELF állományokhoz tartozó segédprogramok alapértelmezés szerint nem teszik láthatóvá a dinamikus linker számára a végrehajtható állományban definiált szimbólumokat. Ennek eredményeképpen a `dlsym()` a `dlopen(NULL, flags)` függvénytõl kapott információk alapján nem találja meg a keresett szimbólumokat.

Ha szükségünk lenne ilyen keresésekre a `dlsym()` használata során a program végrehajtható állományán belül, akkor az adott programot a `--export-dynamic` opció megadásával kell linkelni (lásd ld(1)).

18.15. Hogyan növelhetõ vagy csökkenthetõ a rendszermag címtere i386™ architektúrán?

Az i386™ platformon a rendszermag címtere alapértelmezés szerint 1 GB (PAE esetén 2 GB). Ha komolyabb hálózati forgalmat bonyolító szerverünk van (például egy nagyobb FTP vagy HTTP szerver) vagy rendszerükön használni akarjuk a ZFS állományrendszert, akkor könnyen kifuthatunk a címtérbõl.

A címtér méretének megváltoztatásához vegyük fel a következõ sort a rendszermag konfigurációs állományába, majd fordítsuk újra a rendszermagot:

```
options KVA_PAGES=N
```

Az *N* megfelelõ értékének megállapításához osszuk el a beállítani kívánt címtér (MB-okban megadott) méretét néggyel. (Tehát például 2 GB esetén ez 512 lesz.)

Chapter 19. Köszönetnyilvánítás

Ezt a szegény kis ártatlan GYIKocskát több százan, ha nem is éppen több ezren írták, újraírták, szerkesztették, hajtogatták, tekergették, csonkítgatták, kibelezték, nézegették, összekutyulták, emlegették, felöklendezték, újraépítették, javítgatták és felpezsdítették az utóbbi években. Folyamatosan.

Ezúton is szeretnénk köszönetet mondani mindazoknak, akik gondozásukba vették, és mindenkit csak bátorítani tudunk, hogy csatlakozzon hozzájuk a GYIK továbbfejlesztésében.

Irodalomjegyzék

[biblio-unleashed] FreeBSD Unleashed. Michael Urban und Brian Tiemann. Sams. Erste Ausgabe. 992 Seiten. Oktober 2001. ISBN 0-67232-206-4.

[biblio-44sysman] 4.4BSD System Manager’s Manual. Computer Systems Research Group, University of California, Berkeley. O’Reilly and Associates. Erste Ausgabe. Juni 1994. 804 Seiten. ISBN 1-56592-080-5.

[biblio-44userman] 4.4BSD User’s Reference Manual. Computer Systems Research Group, University of California, Berkeley. O’Reilly and Associates. Erste Ausgabe. Juni 1994. 905 Seiten. ISBN 1-56592-075-9.

[biblio-44suppman] 4.4BSD User’s Supplementary Documents. Computer Systems Research Group, University of California, Berkeley. O’Reilly and Associates. Erste Ausgabe. Juni 1994. 712 Seiten. ISBN 1-56592-076-7.

[biblio-44progman] 4.4BSD Programmer’s Reference Manual. Computer Systems Research Group, University of California, Berkeley. O’Reilly and Associates. Erste Ausgabe. Juni 1994. 866 Seiten. ISBN 1-56592-078-3.

[biblio-44progsupp] 4.4BSD Programmer’s Supplementary Documents. Computer Systems Research Group, University of California, Berkeley. O’Reilly and Associates. Erste Ausgabe. Juni 1994. 596 Seiten. ISBN 1-56592-079-1.

[biblio-44kernel] The Design and Implementation of the 4.4BSD Operating System. M. K. McKusick, Kirk Marshall, Keith Bostic, Michael J Karels und John Quarterman. Addison-Wesley. Reading MA . 1996. ISBN 0-201-54979-4.

[biblio-freebsdkernel] The Design and Implementation of the FreeBSD Operating System. M. K. McKusick und George V. Neville-Neil. Addison-Wesley. Boston MA . 2004. ISBN 0-201-70245-2.

[biblio-nemeth3rd] Unix System Administration Handbook. Evi Nemeth, Garth Snyder, Scott Seebass, Trent R. Hein und John Quarterman. Prentice-Hall. Dritte Ausgabe. 2000. ISBN 0-13-020601-6.

[lehey3rd] The Complete FreeBSD. Greg Lehey. Walnut Creek. Dritte Ausgabe. Juni 1999. 773 Seiten. ISBN 1-57176-246-9.

[McKusick et al, 1994] Berkeley Software Architecture Manual, 4.4BSD Edition. M. K. McKusick, M. J. Karels, S. J. Leffler, W. N. Joy und R. S. Faber. 5:1-42.

[biblio-ja-fbsdpc98] FreeBSD for PC 98’ers (in Japanisch). SHUWA System Co, LTD.. ISBN 4-87966-468-5 C3055 P2900E.

[biblio-ja-fbsd] FreeBSD (in Japanisch). CUTT. ISBN 4-906391-22-2.

[biblio-ja-compintro] Complete Introduction to FreeBSD (in Japanisch). Shoeisha Co., Ltd. ISBN 4-88135-473-6 P3600E.

[biblio-ja-unixstarterkit] Personal UNIX Starter Kit FreeBSD (in Japanisch). ASCII. ISBN 4-7561-1733-3 P3000E.

[biblio-ja-fbsdhb] FreeBSD Handbook (Japanische Übersetzung). ASCII. ISBN 4-7561-1580-2 P3800E.

[biblio-ge-fbsdmitmeth] FreeBSD mit Methode (in Deutsch). Computer und Literature Verlag/Vertrieb Hanser. 1998. ISBN 3-932311-31-0.

[biblio-ja-fbsdinstandutil] FreeBSD install and Utilization Manual (in Japanisch). Mainichi Communications Inc..

[biblio-indo-intserv] Building Internet Server with FreeBSD (in Indonesisch). Elex Media Komputindo. Onno W Purbo, Dodi Maryanto, Syahrial Hubbany und Widjil Widodo.

[biblio-fbsdcorpnetguide] The FreeBSD Corporate Networker's Guide. Addison-Wesley.

[biblio-unixnutshell] UNIX in a Nutshell. O'Reilly & Associates, Inc.. 1990. ISBN 093717520X.

[biblio-cantfindadmin] What You Need To Know When You Can't Find Your Unix System Administrator. O'Reilly & Associates, Inc.. 1995. Linda Mui. ISBN 1-56592-104-6.

[biblio-ja-fbsdusrrefman] FreeBSD User's Reference Manual (Japanische Übersetzung). Mainichi Communications Inc.. Jpman Project, Japan FreeBSD Users Group. 1998. ISBN 4-8399-0088-4 P3800E.

[biblio-newcomeunix] Online Guide for newcomers to the UNIX environment". Edinburgh University.

[biblio-dnsandbind] DNS and BIND. O'Reilly & Associates, Inc. ISBN 1-56592-512-2. Paul Albitz Albitz und Cricket Liu. 1998. Dritte Ausgabe.

[biblio-sendmail] Sendmail. O'Reilly & Associates, Inc. 1997. Zweite Auflage. Brian Costales. ISBN 1-56592-222-0.

[biblio-esssysadmin] Essential System Administration. Æleen Frisch. Zweite Auflage. O'Reilly & Associates. 1995. ISBN 1-56592-127-5.

[biblio-tcpipnetworkadministration] TCP/IP Network Administration. Craig Hunt. Zweite Auflage. O'Reilly & Associates, Inc. 1997. ISBN 1-56592-322-7.

[biblio-managingnfsandnis] Managing NFS and NIS. Hal Stern. O'Reilly & Associates, Inc. 1991. ISBN 0-937175-75-7.

[biblio-jpmanprojectjfug] FreeBSD System Administration's Manual. Jpman Project, Japan FreeBSD Users Group. Mainichi Communications Inc.. 1998. ISBN 4-8399-0109-0 P3300E.

[biblio-xwinsystoolkit] X Window System Toolkit. Digital Press. Paul Asente. ISBN 1-55558-051-3.

[biblio-carefman] C: A Reference Manual. Prentice Hall. 1995. Vierte Auflage. Samuel P. Harbison und Guy L. Jr. Steele. ISBN 0-13-326224-3.

[biblio-thecproglang] The C Programming Language. Prentice Hall. 1998. Brian Kernighan und Dennis Ritchie. ISBN 0-13-110362-9.

[biblio-portingunixsoft] Porting UNIX Software. Greg Lehey. O'Reilly & Associates, Inc.. 1995. ISBN 1-56592-126-7.

[biblio-thestandardclibrary] The Standard C Library. Prentice Hall. 1992. P. J. Plauger. ISBN 0-13-131509-9.

[biblio-advprogintheunixenv] Advanced Programming in the UNIX Environment. Addison-Wesley. 1992. W. Richard Stevens. ISBN 0-201-56317-7.

[biblio-unixnetprog] UNIX Network Programming. W. Richard Stevens. Prentice Hall. 1998. Zweite Auflage. ISBN 0-13-490012-X.

[biblio-writeserialdriverforunix] Writing Serial Drivers for UNIX. Bill Wells. Dezember 1994. Dr. Dobb's Journal. pp68-71, pp97-99.

[biblio-unixsysarch] UNIX System Architecture. Prentice-Hall, Inc. 1990. Prabhat K. Andleigh. ISBN 0-13-949843-5.

[biblio-portingunixtothe386] Porting UNIX to the 386. William Jolitz. Dr. Dobb's Journal. Januar 1991 - Juli 1992.

[biblio-tcpipillv1theprotocols] TCP/IP Illustrated, Volume 1: The Protocols. W. Richard Stevens. Addison-Wesley. 1996. ISBN 0-201-63346-9.

[biblio-unixsysformodrnarch] Unix Systems for Modern Architectures. Addison-Wesley. Curt Schimmel. 1994. ISBN 0-201-63338-8.

[biblio-tcpipillvol3] TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP and the UNIX Domain Protocols. Addison-Wesley. 1996. W. Richard Stevens. ISBN 0-201-63495-3.

[biblio-unixinternthenewfrontiers] UNIX Internals — The New Frontiers. Uresh Vahalia. Prentice Hall. 1996. ISBN 0-13-101908-2.

[biblio-tcpipillvol2theimplementation] TCP/IP Illustrated, Volume 2: The Implementation. Gary R. Wright und W. Richard Stevens. 1995. Addison-Wesley. ISBN 0-201-63354-X.

[biblio-firewallsandinternetsecurity] Firewalls and Internet Security: Repelling the Wily Hacker. William R. CHeswick und Steven M. Bellovin. Addison-Wesley. 1995. ISBN 0-201-63357-4.

[biblio-practicalunixsecurity] Practical UNIX Security. Simson Garfinkel und Gene Spafford. 1996. Zweite Auflage. O'Reilly & Associates, Inc. ISBN 1-56592-148-8.

[biblio-pgpprettygoodprivacy] PGP Pretty Good Privacy. Simson Garfinkel. O'Reilly & Associates, Inc. 1995. ISBN 1-56592-098-8.

[biblio-pentiumprocarch] Pentium Processor System Architecture. Don Anderson und Tom Shanley. Addison-Wesley. 1995. Zweite Auflage. ISBN 0-201-40992-5.

[biblio-progguidetothesvgacards] Programmer's Guide to the EGA, VGA, and Super VGA Cards. Richard F. Ferraro. Dritte Ausgabe. Addison-Wesley. 1995. ISBN 0-201-62490-7.

[biblio-80486] 80486 System Architecture. Tom Shanley. Addison-Wesley. 1995. Dritte Ausgabe. ISBN

0-201-40994-1.

[biblio-isasysarch] ISA System Architecture. Tom Shanley. Addison-Wesley. Dritte Ausgabe. 1995. ISBN 0-201-40996-8.

[biblio-pcisysarch] PCI System Architecture. Tom Shanley. Addison-Wesley. 1995. Dritte Ausgabe. ISBN 0-201-40993-3.

[biblio-theundocumentedpc] The Undocumented PC. Frank Van Gilluwe. Addison-Wesley. 1994. ISBN 0-201-62277-7.

[biblio-bellsystemtechnicaljournal] Bell System Technical Journal, Unix Time-Sharing System. American Telephone & Telegraph Company. Juli - August 1978. Vol 57, No 6, Part 2. ISSN0005-8580.

[biblio-commentaryonunix] Lion’s Commentary on UNIX. John Lion. ITP Media Group. 1996. Sechste Ausgabe. ISBN 1573980137.

[biblio-newhackerdict] The New Hacker’s Dictionary. Eric S. Raymond. MIT Press. 1996. Dritte Ausgabe. ISBN 0-262-68092-0.

[biblio-aqtrcentofunix] A quarter century of UNIX. Peter H. Salus. Addison-Wesley. 1994. ISBN 0-201-54777-5.

[biblio-unixhatershandbook] The UNIX-HATERS Handbook. Steven Strassman, Daniel Weise und Simon Garfinkel. IDG Books Worldwide, Inc. 1994. ISBN 1-56884-203-1.

[biblio-lifewithunix] Life with UNIX - special edition. Don Libes und Sandy Ressler. Prentice-Hall. 1989. ISBN 0-13-536657-7.

[biblio-bsdfamilytree] The BSD Family Tree. 1997.

[absolutebsd] Absolute BSD. Michael Lucas. No Starch Press. Juni 2002. ISBN 1-886411-74-3.

[biblio-ccppusersjournal] The C/C++ Users Journal. R&D Publications Inc.. ISSN 1075-2838.

[biblio-sysadminthejournalforunixsysadmins] Sys Admin - The Journal for UNIX System Administrators. Miller Freeman, Inc. ISSN 1061-2688.