FLOSSzine

Free / Libre / Open Source Software fanzine

2009. július

II. évfolyam 2. szám

VoIP Linux alatt

Bos Wars

Háború, Linuxon

Hálózatbiztonság

nyílt forrású eszközökkel

http://flosszine.org

Tartalom

Lite

Pro

Tisztelt Olvasó!

Csaknem napra pontosan egy évvel ezelőtt jelent meg a FLOSSzine elektronikus úton terjesztett fanzin első lapszáma. Az Olvasó most a soron következő ötödiket tartja virtuálisan (illetve, ha kinyomtatta, akkor ténylegesen) a kezében, ami a II. évfolyam 2. lapszáma.

Akkor még csak remélni mertük, hogy megérjük az egy éves születésnapot, ugyanis a 2008-as februári „tagtoborzó felhívás" kapcsán nem sok biztatóval kecsegtettek a hasonló körökben jártasabbak. Egészen pontosan „nem adtak nekünk 1-2 lapszámnál többet". Ehhez képest sikerült megvalósítani, amit többek eleve lehetetlennek tartottak. 0 Ft támogatással (pedig kiadásaink azok vannak), teljesen önerőből, mondhatni hobbiból és szabadidőben, a nyílt forráskódú és szabad szoftverek melletti elkötelezettségtől vezérelve úgy fennmaradni, hogy büszkén állíthatjuk, hogy még létezünk és eszünk ágában sincs abbahagyni. Ha valamilyen oknál fogva mi mégsem tudnánk folytatni (aminek jelen állás szerint csekély az esélye), akkor biztosan megtaláljuk azokat az önkénteseket akik tovább tudják vinni a stafétát.
Most, hogy eltelt egy kis idő, ideje számot adnom meglévő és leendő Olvasóink fele. Mit értünk el – és mit nem – ezalatt az esztendő alatt?

Beszéljenek a tények helyettem:

- *Interjúkat készítettünk: Baja Ferenc (Miniszterelnöki Hivatal), Balsai Péter (Szabad Szoftver Intézet), Keszei Balázs és Kőnig Tibor (Microsoft), Kürti László (Open Source Farm), Micskó Gábor (hup.hu), Nagy Róbert (OpenBSD), Németh László (Hunspell), Richard M. Stallman (GNU), Scheidler Balázs (syslog-ng), Dr. Szentiványi Gábor (Linux Ipari Szövetség)*
- *Portrékat írtunk a szabad szoftveres mozgalom meghatározó személyiségeiről: Alan Cox, Erik S. Raymond, Jon „maddog" Hall, Richard M. Stallman*
- *Programozási ismereteket nyújtottunk: bash, C, C++, GTK+, gtkmm, Makefile (Olvasóink közül, aki akart, belemélyedhetett lapszámainkat forgatva ezen programnyelvek megismerésébe).*
- *Partnerkapcsolatokat építettünk: Free Libre Open Source Software Farm, Frugalware Linux, Linux Akadémia, fsf.hu Alapítvány, Linux Felhasználók Magyarországi Egyesülete, Magyar Ubuntu Közösség, SKL Projekt*

Statisztikák:

- *5 lapszám, 60 cikk, 270 oldal, 23,1 MB adat*
- *3000 (stabil) olvasó lapszámonként (meggyőződésünk, hogy a mérhető adatokon túl még szépszámú Olvasótáborunk van)*
- *10 588 egyedi látogató, 47 746 oldalmegtekintés, 158 regisztrált felhasználó (a letöltéshez nem, csak a hozzászóláshoz szükséges a regisztráció)*
- *11 stabil szerkesztőségi tag*
- *690 (e-mail) üzenet a belső levelezőlistánkon + 434 üzenet a vezetőség listáján*

Amire mindeddig kisebb hangsúlyt fektettünk (mert először önmagunk előtt is bizonyítani akartunk): önmagunk hirdetése, közösségépítés. Ezentúl igyekszünk ezen hiányosságokat pótolni. Olvasóink a honlapunkon leadott szavazataikkal ezután is befolyásolhatják, hogy miből legyen több, és miből kevesebb a lapban, ahogyan e-mailben a részletesebben kifejtett ötleteiket is várjuk. Azonban természetesen ötletekkel mi magunk is tele vagyunk, úgyhogy még ennél is hathatósabban segíti a FLOSSzine-t mindenki, aki elküldi írásait a szerkesztőségünknek.

További információk:

http://www.FLOSSzine.org
http://en.wikipedia.org/wiki/FLOSS
http://en.wikipedia.org/wiki/Fanzine

Szerző és szerkesztőtársaim nevében is tisztelettel és köszönettel:
Horváth Örs Apor
alapító-főszerkesztő
2009. július 17., Budapest

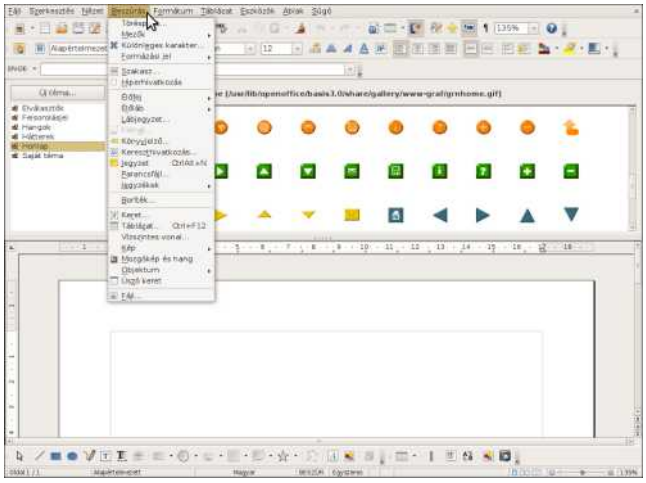
Writer Calc Base Impress Draw Math

Szövegszerkesztés szabad szoftverekkel

Vannak feladatok, amiket ma már egyre nehezebb elképzelni számítógépes eszközök nélkül. Ezek egyike a szövegszerkesztés, amellyel kapcsolatban közkeletű az a tévhit, hogy meg kell tanulni egy programot. Ennél sokkal célszerűbb a szövegszerkesztés elvét megtanulni, hiszen annak ismeretében már az összes szövegszerkesztő program elvével is jó eséllyel tisztában leszünk. Emellett a módszer mellett szól az is, hogy a programok úgyis változnak, fejlődnek, de mi magunk is szembe kerülhetünk más szoftverkörnyezettel, amikor jó, ha gyorsan kiismerjük magunkat.

Az elv

Az emberek egy része már nagyon régóta hoz létre és szerkeszt szöveget, de kevesen gondolták végig, hogy mi is történik a folyamat során. Először is a szöveget elő kell állítani, le kell írni. Már ez is sokféleképpen lehetséges: füstjelekkel, rovásírással egy botra, kőbe véséssel, vagy esetleg papiruszra írással. Jó lenne a kész szöveget eltárolni, amennyiben ez szükséges (általában igen), no itt a füst kiesik – hacsak le nem fényképezzük, de a füstjel és a fényképezőgép egyidejű jelenléte nem jellemző. A kőtáblák és a botok kezelése már önmagában is igen nehézkes, főképp ha a szöveg igényelné a tagolást, vagy az átszerkesztést. Itt már erősen előnyben van a papír-ceruza-radír triumvirátus és olyan kapcsolt eszközök, mint a korrektorfesték. Ez vonatkozik a mechanikus és elektromechanikus írógépekre is. Még a sokszorosítás és a másolás is megoldható a hagyományos eszközök egy részével, ide értve a mechanikus nyomdatechnikát is, bár az jóval körülményesebb eljárás, mint amit a mai modern technológia nyújt. Ezért történt, hogy az elektronikus szövegszerkesztő masinák rohamosan elterjedtek, és mára szinte egyeduralkodóvá vált a számítógépes szövegszerkesztés.

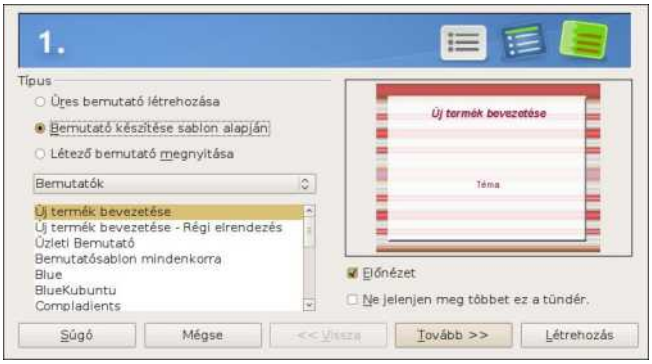


Az ár

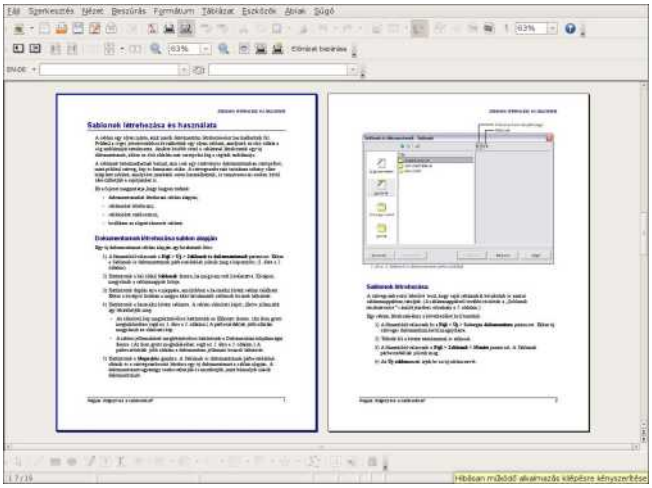
Nos, a számítógép ára ma még mindenképpen beletartozik a költségekbe, ahogyan a járulékos dolgok is, mint az áram, vagy a festékkazetta. A szoftverek terén már más a helyzet: akár az alaprendszer is lehet teljesen ingyenes. A nem ingyenes operációs rendszerekre is lehet komoly ingyenes, vagy nyílt forrású rendszereket találni, de nézzük a leginkább költségmentes megoldást.

A gépre kell valamilyen operációs rendszer és alapesetben más semmi. Választani van miből, legfeljebb nehéz. Segít ebben a *DistroWatch* nevű oldal, amely naprakész információkat közöl a nyílt forráskódú rendszerekről. A legtöbb disztribúció feltelepítve azonnal több lehetőséget is kínál a szövegek szerkesztésére, a kisebb editoroktól kezdve a komoly irodai rendszerekig. Korábban jobban elkülönült a sima szövegszerkesztés és a kiadványszerkesztés, de ma már egyre inkább a mindent bele megoldásokat preferálják a gyártók, bár jelenleg is jelentős különbségek vannak a különböző célú programok között. Nem véletlenül használnak kiadványszerkesztő rendszereket azok, akiknek ez a munkájuk. Szerkeszthetünk szöveget akár soreditorral is, vagy a rendszer programszerkesztőivel is, a szövegszerkesztési munkák legnagyobb része ezekkel is elvégezhető. Persze egy soreditornál (amit már a shell parancssori környezet is biztosít számunkra, jobban használható a `vi` (ma már ViM) szerkesztő.

Amennyiben vesszük a fáradtságot és megnézzük a mai rendszerek sem képesek igazán másra, mint a korábban említettek, csak a feladatokat könnyebben és számunkra kényelmesebben teljesítik.



Impress, a bemutatókészítő



Munkában a Writer

A választás

Komolyabb feladatok esetén is rengeteg lehetőségünk van a választásra. Vegyük sorra! A legismertebb és valószínűleg a legtöbb funkciót nyújtó az ingyenesek közül az *OpenOffice.org* nevezetű irodai rendszer, ami már régen több, mint egy egyszerű szövegszerkesztő. Felveszi a versenyt a fizetős alkalmazásokkal, sőt bizonyos területeken többet tud náluk. Az OpenOffice.org (és különféle mutációi, például az *OxygenOffice Professional*) szinte minden szükséges alkalmazást biztosít számunkra, amire csak szükséges lehet az irodai munkák során:

Writer – ez maga a szövegszerkesztő

Calc – az OpenOffice.org táblázatkezelője

Base – az adatbázis-kezelő

Impress – a bemutatók készítéséhez használható program

Draw – a rajzolóprogram

Math – egy közönséges szövegszerkesztőben nem könnyű (és főleg nem látványos) bonyolult matematikai képleteket beírni, de az *OpenOffice.org* esetében más a helyzet. Ezt a feladatot könnyíti meg a Math modul.

Az OpenOffice.org már régóta tud PDF (Portable Document Format) állományokat is készíteni, valamint simán olvassa és írja a Microsoft Office formátumait, ami fordítva többségében mondható el (a Microsoft Office 2007 SP2-től kezdve – bár nem teljeskőrűen – de támogatja a funkciót). A kompatibilitási problémákat felrovóknak igazuk van, csak általában azt felejtik el, hogy a Microsoft Office kiadások sokszor egymással sem kompatibilisek, valamint egy bonyolultabb fájl gyakran még a szülőalkalmazáson is kifog, hiszen a Word nem kiadványszerkesztő, az Excel pedig nem vezetői információs, vagy főkönyvi rendszer. Sőt: a tapasztalat azt mutatja, hogy sokszor egy sérült Microsoft Office állományt, amit a saját szülőalkalmazása sem tud megnyitni, vagy helyesen megjeleníteni (hanem inkább kómába esik), még meg tud menteni az OpenOffice.org, ha a megfelelő modulját alkalmazzuk.

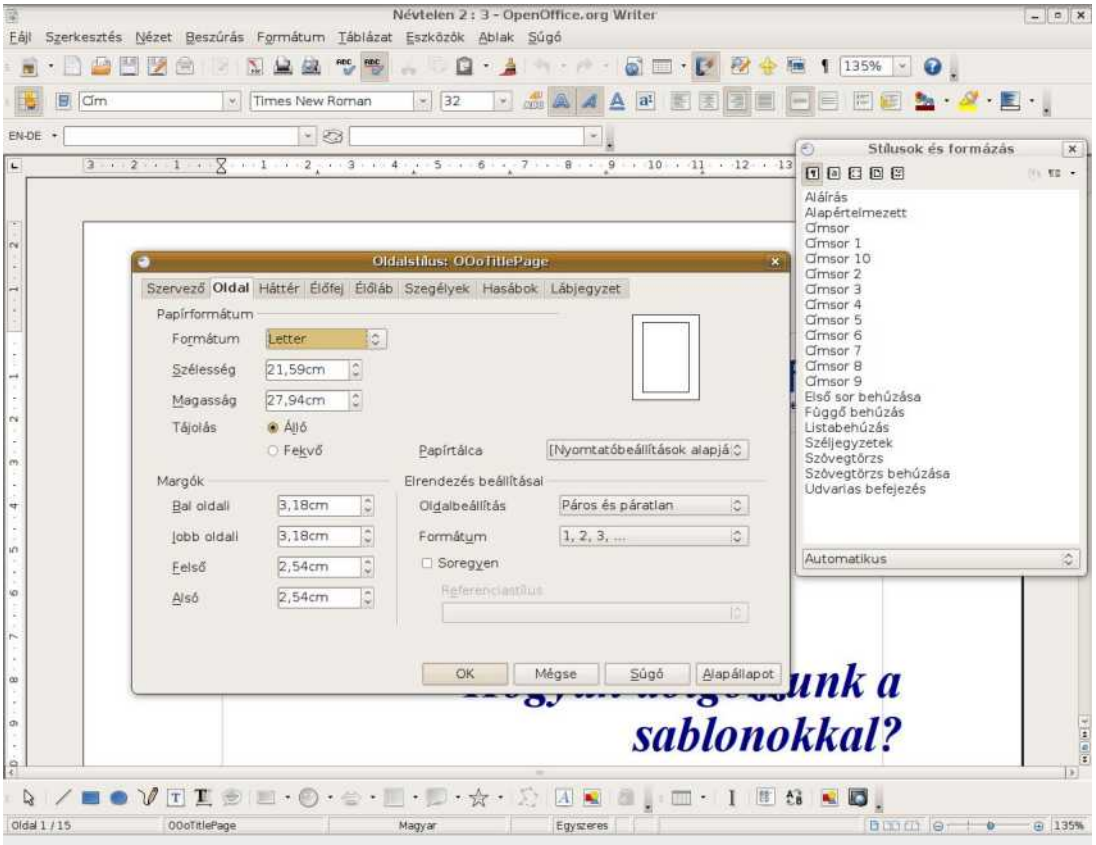
Más választás

A *Solaris* és *OpenSolaris* alkalmazásait is használhatjuk irodai célokra, ugyan a legfrissebb StarOffice rendszerek fizetősek, de a régebbi verziók ingyen letölthetőek az alaprendszerrel együtt. A SourceForge.net-en keresgélve megint csak a bőség zavarával vagyunk kénytelenek megküzdeni, és valószínűleg nem mi fogunk győzni, bátran futamodjunk meg, hiszen az itt felsorolt programok kipróbálása, tesztelése nem szöveget szerkeszteni akaró emberhez mért feladat. Persze itt megint csak hagyatkozhatunk a közösség erejére, a legjobb programokról rengeteg információt találhatunk a közösségi oldalakon és bízhatunk a mértéktartóan megfogalmazott véleményekben. Annyi kivehető a nagy keresgélésből, hogy a szövegszerkesztő programok világa is erősen szegmentálódott mára. A különböző célfeladatokra a megfelelő céleszközök használata a megoldás (jegyzettömbök, programozói editorok, szövegszerkesztő programok, kiadványszerkesztő rendszerek, stb.).

A különböző disztribúciók az elterjedt irodai rendszerek mellett gyakran felkínálnak más, komoly alkalmazásokat is, pl.: *KOffice*, ezekről se feledkezzünk meg. A nagy irodai rendszereken kívül érdemes kipróbálni az alábbi két programot is:

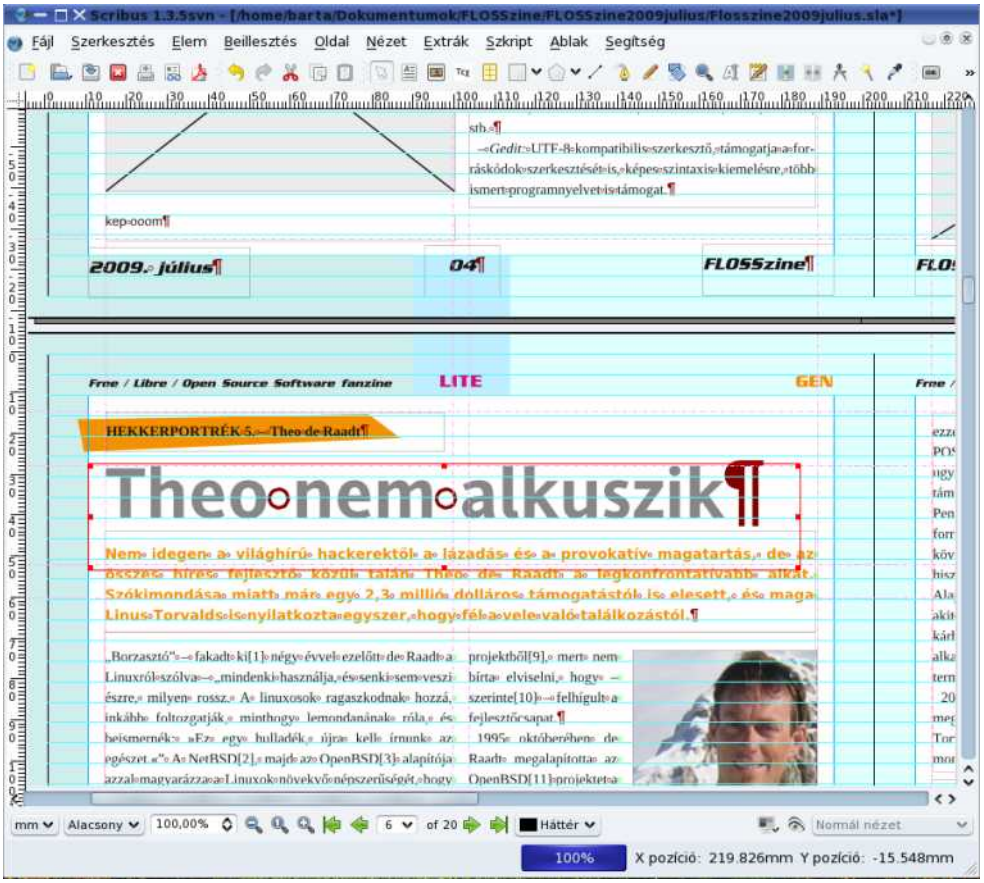
– *AbiWord:* remekül használható multiplatformos szövegszerkesztő (Windows, Linux, QNX, FreeBSD, Solaris alá telepíthető), kezeli a legismertebb dokumentumformátumokat: ODT, Microsoft Word, WordPerfect, RTF, HTML, stb.

– *Gedit:* UTF-8 kompatibilis szerkesztő, támogatja a forráskódok szerkesztését is, képes szintaxis kiemelésre, több ismert programnyelvet is támogat.



Oldalbeállítás és Stílusok ablakok a Writer-ben

Készül a FLOSSzine magazin a Scribus legújabb verziójával



Hasonlóságok

Láthatjuk, hogy a szerkesztőprogramok működhetnek parancssorból, használhatnak szöveges képernyőt vagy grafikus felületet. Támogathatják a fejlett szövegszerkesztői funkciókat, vagy vannak, amelyek csak egy jegyzettömböt helyettesítenek. Alkalmazhatók forráskód szerkesztésére, vagy komolyabb kiadványok készítésére, mégis nagyon hasonlítanak egymásra a lényeges dolgokban. Mindegyikben megtalálható minden szükséges alapfunkció a szöveg előállításához. Ilyenek ezek a menüpontok:

New/Új – elővesszük a lapot és teleírjuk

Save/Mentés – eltesszük valahová

Open/Megnyit – megnyitjuk, elővesszük (szerkeszthetjük a szöveget újra).

A *Print* parancsra a dokumentum papír alapú megjelenítése miatt van szükség, mivel a szöveg szerkesztése és megjelenítése elvált egymástól a digitális eszközöket használva. Az összes többi menüpont már csak kiterjeszti lehetőségeinket a szerkesztési feladatok megoldása során.

Persze a határterületeken, és a kiegészítő alkalmazások között is nagyon sok érdekes programot találhatunk. Akinek kiadványszerkesztőre van szüksége, az nyugodtan próbálkozhat a *Scribus*-szal, jelenleg magazinunk is ebben nyeri el végső formáját.

Képfunkciók

Főleg informatikai, műszaki kiadványok elkészítésénél gyakran lehet szükség a képernyőképekre. Ezt remekül megoldhatjuk szabad szoftveres megoldásokkal. Ugyancsak fontos lehet a különböző állományok (itt elsősorban képfájlokra gondolok) átalakítása, konvertálása, erre szintén találunk igen kiváló szabad programokat. Képernyőképek előállítására az open source alaprendszerek is kínálnak megfelelő megoldásokat, mind a GNOME (`gnome-screenshot`), mind a KDE (*KSnapshot*) grafikus környezetet használva. A legfrissebb disztribúcióknál már csak elég csak megnyomni a Print screen billentyűt az aktuális képernyőkép elkészítéséhez.

A *Gimp* szintén mindkét funkciót tudja (screenshot, konvertálás), ahogy az *ImageMagic* nevű alkalmazás is, amelyik több mint száz formátumot ismer. Telepítése a szokásos módon zajlik (a rendszerünket előtte érdemes frissítenünk, update után már szó nélkül feltelepül. A képernyőkép elkészítését időzíthetjük is, a konvertálás pedig rendkívül egyszerűen a parancssorból kezdeményezhető.

Az *ImageMagic* kezeli a text állományokat is. Visszafelé is működik, de használata során alapesetben nem ASCII grafikát fogunk kapni. (Így ezt csak akkor ajánljuk, ha rengeteg információra van szükség az adott képről). Néha szükség lehet még a konzol képernyő mentésére is, ennek egyik megoldása a:

`cat /dev/vcsN > fájlnév` karaktersorozat, ahol N a lementendő tty száma.

Mára ennyit a szövegszerkesztés világából. Legközelebb a könyvelő álmával folytatjuk, vagyis a táblázatkezelő, vagy számolótábla programokról ejtünk néhány szót.

Szőke József

Tudtad-e?

Online böngészőteszt

Négyféle operációs rendszer alatt futó - mintegy 17-féle böngésző - összesen több mint 100 különböző verziójával próbálhatjuk ki, hogyan jelenik meg az adott konfigurációval egy adott weboldal - ha használjuk a BrowserShots[1] nyílt forráskódú online szolgáltatást. Vagy fordítva: ha különböző, számunkra fontos weboldalakhoz, illetve hálózati funkciókhoz keressük éppen az operációs rendszerünk alatt legjobban működő böngészőt, akkor is érdemes ezt a szolgáltatást kipróbálni. A honlapkészítők e kézre álló eszközének csupán annyi a hiányossága, hogy egyszerre túl sok megjelenítést nem érdemes indítani vele, mert órákba is telhet, amíg a BrowserShots kiadja a böngészőképeket.

[1] http://browsershots.org/

HEKKERPORTRÉK 5. – Theo de Raadt

Theo nem alkuszik

Nem idegen a világhírű hackerektől a lázadás és a provokatív magatartás, de az összes híres fejlesztő közül talán Theo de Raadt a legkonfrontatívabb alkat. Szókimondása miatt már egy 2,3 millió dolláros támogatástól is elesett, és maga Linus Torvalds is nyilatkozta egyszer, hogy fél a vele való találkozástól.

„Borzasztó” – fakadt ki[1] négy évvel ezelőtt de Raadt a Linuxról szólva – „mindenki használja, és senki sem veszi észre, milyen rossz. A linuxosok ragaszkodnak hozzá, inkább foltozgatják, minthogy lemondanának róla, és beismernék: »Ez egy hulladék, újra kell írnunk az egészet.«” A NetBSD[2], majd az OpenBSD[3] alapítója azzal magyarázza a Linuxok növekvő népszerűségét, hogy szerinte a Linux-kernel fejlesztői lepaktáltak az olyan nagy hardvergyártókkal, mint a HP, vagy az IBM. De Raadt szerint a gyártók pedig jobban jártak ezzel, mint ha továbbra is csak a zárt fejlesztésű drivereikkel adnák el a termékeiket, hiszen így ingyenes munkaerőre támaszkodhatnak.

Theo de Raadt[4] a dél-afrikai Pretóriában született 1968-ban, holland apától és dél-afrikai anyától. A család, amely később három fiatalabb testvérével is gyarapodott, 1977-ben áttelepült Kanadába, az albertai Calgaryba[5]. Egy Yukon állam-beli kitérő után de Raadt a Calgary Egyetemen szerzett szoftvermérnöki diplomát és ma is a hosszú, száraz teleiről ismert városban él, ahol a tomboló nyárban is csak ritkán emelkedik 20 fok fölé a hőmérséklet. Theo, Nadine nevű barátnőjével, valamint a Galilei és a Kepler nevezetű macskákkal osztja meg lakhelyét és az OpenBSD-n kívül sörfőzéssel, hegymászással, barlangászással is foglalkozik, valamint a mountainbike-ozást kedveli[6].

Ő alapította a NetBSD-t[7], Chris Demetriou, Adam Glass és Charles Hannum társaságában. Az a cél vezérelte őket, hogy a Berkeley Egyetemen[8] fejlesztett operációs rendszert nyílttá, szabadon fejleszthetővé, több platformúvá, és hordozhatóvá tegyék, illetve, hogy a foltoktól megtisztított kód segítségével egy gyártható minőségű, BSD-n alapuló operációs rendszert fejlesszenek ki. A rendszert eredetileg a BSD 4.3-as kiadásának a kódtisztításával indították el. Az első hivatalos kiadást, a NetBSD 0.8-ast, 1993-ban tették közzé, az azonban már a BSD 386-osnak a 0.1-es verziószámú forráskódját, valamint a 0.2.2 jelű, nem hivatalos foltot tartalmazta. Az 1.0-ás a következő évben jelent meg, ám Theo de Raadt néhány hónap múlva – heves viták kíséretében – kivált a projektből[9], mert nem bírta elviselni, hogy – szerinte[10] – felhígult a fejlesztőcsapat.

Theo de Raadt: hegyet is mászik

Mindent a biztonságért…

1995 októberében de Raadt megalapította az OpenBSD[11] projektet a NetBSD 1.0-ásra alapozva. 1996-tól fél évente követik egymást a rendszer[12] új kiadásai, az utolsó a 2009. május 1-én megjelent 4.5-ös[13]. Az OpenBSD-t a hordozhatóság, a szabványosság, a helyes működés, a megelőzésen alapuló biztonság és a beépített titkosítás optimalizálásának a jegyében fejlesztik. Két évvel ezelőtt hívták életre az OpenBSD alapítványt, amelyen keresztül hivatalosan is támogatni lehet a projekt által kínált összes terméket, így az OpenSSH-t, az OpenBGPD-t, az OpenCVS-t és OpenNTPD-t is. A DistroWatch statisztikája[14] szerint az OpenBSD az elmúlt 12 havi Unix/Linux népszerűségi listán a 47. helyet foglalja el, a 16. helyen álló FreeBSD, és a 23. helyet elfoglaló PC-BSD mögött. A 75. helyezett NetBSD-t még a DesktopBSD is megelőzi, amelyik az 52. a rangsorban.

Theo 2003 áprilisában szerencsétlenségére kifejtette lesújtó véleményét az Egyesült Államok iraki beavatkozásáról és ezzel kútba esett az OpenBSD részvételével tervezett

POSSE-projekt[15]. Az amerikai védelmi minisztérium ugyanis rögtön törölte[16] azt a 2,3 millió dolláros támogatást, amelyet a projektgazdának jelentkező Pennsylvaniai Egyetemnek adott volna a hordozható, nyílt forráskódú biztonsági rendszerek kifejlesztésére. A következő év azonban jobban sikerült de Raadt számára, hiszen személyesen Richard Staalmanntől vehette át az FSF Alapítvány Szabad Szoftver Díját[17], attól az RMS-től, akit egyébként még NetBSD-alapító korában azért kárhoztatott, hogy ugyan „szabadnak” nevezi az alkalmazásait, de kódjait nem tisztítja meg a kereskedelmi termékek érdekében végzett foltozásoktól.

2005-ben aztán az amerikai Forbes Magazinban nyíltan megtámadta[18] a Linux-kernel megalkotóját, Linus Torvaldsot és általában a linuxos fejlesztőközösségeket, mondván, hogy az általuk kitalált fejlesztési modell minden visszásság oka. Szerinte folyamatosan rontja a kód minőségét az, hogy az egyes programok karbantartói (maintainer) bizonyos részleteket küldözgetnek Linusnak, illetve néhány kiválasztott fejlesztőnek. Ezzel szemben az ő fejlesztőcsapata mindössze 60, szorosan együttműködő programozóból áll, nyilatkozta, és a „valódi Unix kód” minőségét tartja szem előtt. De Raadt hozzátette: a két modell közötti legnagyobb különbséget a motivációban látja. Míg a Linux esetében csupán az olcsó hackekkel elért működőképesség a cél, addig az OpenBSD-nél a kód minősége elsődleges, mondta. A linuxosok nagy zajjal járó Microsoft-ellenessége dacára szerinte a két rendszer sok tekintetben egyre jobban hasonlít egymásra. Theo ilyennek tartja például az általában túl rövidre szabott kiadási időszakokat, amelyeknek az az eredménye, hogy romhalmazokat adnak ki és terjesztenek.

Jankovich Oszkár

[1] http://www.forbes.com/2005/06/16/linux-bsd-unix-cz_dl_0616theo.html
[2] http://www.netbsd.org/
[3] http://www.openbsd.org/hu/
[4] http://en.wikipedia.org/wiki/Theo_de_Raadt#cite_note-0
[5] http://www.theage.com.au/articles/2004/10/07/1097089476287.html
[6] http://www.theos.com/deraadt/
[7] http://en.wikipedia.org/wiki/NetBSD
[8] http://en.wikipedia.org/wiki/Computer_Systems_Research_Group
[9] http://mail-index.netbsd.org/netbsd-users/1994/12/23/0000.html
[10] http://www.theage.com.au/articles/2004/10/07/1097089476287.html
[11] http://hu.wikipedia.org/wiki/OpenBSD
[12] http://www.openbsd.org/hu/
[13] http://marc.info/?l=openbsd-announce&m=124111361304488&w=2
[14] http://distrowatch.com/stats.php?section=popularity
[15] http://en.wikipedia.org/wiki/POSSE_project
[16] http://www.workers.org/ww/2003/darpa0501.php
[17] http://en.wikipedia.org/wiki/FSF_Free_Software_Awards
[18] http://www.forbes.com/2005/06/16/linux-bsd-unix-cz_dl_0616theo.html

Linux-nyomok az oktatásban

LOK'n roll

Május idusa az informatika-érettségi ideje. Nyilvános adatok híján egyelőre csak képzeletben követhető, hogy hány iskolában hány tanuló érettségizik éppen nyílt forráskódú alkalmazások, rendszerek és egyáltalán szabad szoftverek segítségével. A választás lehetősége mindenesre adott. De inkább csak elvileg. Erről is szó van, már hetedik éve, minden LOK-rendezvényen, és így volt ez áprilisban is.

Hogy mit keres a Linux az oktatásban, azon lehet vitatkozni. A kérdésre legalább három válasz adható:

1. A GNU/Linuxot oktatják a középiskolákban, mert a számítógép-használathoz és emelt szinten a programozáshoz ma már szükséges, illetve a közeljövőben belátható módon szükséges lesz az így megszerezhető informatikai tudás alkalmazása – bármely munkaterületen.

2. A GNU/Linuxot használják az iskolákban, mert az a legtakarékosabb informatikai szoftveres eszköz a tananyag szemléltetésére és az adminisztratív feladatok elvégzésre is. Felhasználhatnák az iskolák szabadon, ingyen és – talán csak a testnevelést kivéve – valamennyi tantárgy oktatásában, ahogyan természetesen a teljes iskolai, intézményi nyilvántartási rendszer is kezelhető lenne vele.

3. A GNU/Linuxot használják oktatásra az iskolán kívül is, mert hatékonyabb online kapcsolat iskola, tanár, tanuló és szülő között jelenlegi tudásunk szerint nem létezik. Ezt is megtehetnék – szabadon és ingyen – az oktatási intézmények, vagyis kihasználhatnák (egyenként és közösen is) a távoktatási rendszerekben rejlő lehetőségeket, a tananyag online hozzáférhetővé tételétől kezdve a házi feladatok online kiadásán és ellenőrzésén keresztül a tantárgyi versenyek kiértékeléséig, vagy a szülőkkel való intenzívebb kapcsolattartásig.

A három lehetőséget vizsgálva megállapítható, hogy – bár az indokok megalapozottak – a gyakorlat nem felel meg az állításoknak. Vagyis: noha a magyarországi szervereken is és az asztali gépeken is egyre gyorsul a Unixos/Linuxos megoldásokra történő váltás üteme, ez még mindig nem elég ahhoz, hogy a középiskolai tantervek többsége a rendszer működéséről szóló tananyagot tartalmazza. Ebből következően az áhított hazai tudásipar egyik alapvető összetevője még évtizedekig hiányozni fog, mert a döntéshozók nem – vagy csak későn – ismerték fel a szakanyag középfokú oktatásának a szükségességét – legyen szó bármely döntéshozatali szintről, a konkrét iskola szakoktatóitól (az informatika tanároktól) és igazgatójától kezdve, az önkormányzati felügyeleti szervek illetékesein keresztül, a szakmai irányítást vezető minisztériumig, kormányzati titkárságig. Ezek a döntéshozók (tisztelet a ritka kivételnek) ma is abból indulnak ki, amit a személyi számítógépeken futó programok piacán látnak, tehát egy piacvezető operációs rendszer hegemónia-közeli elterjedtségéből. Ennek oka kétféle lehet: egyrészt az, hogy maguk is csak azt használják, így felhasználóként is csak azzal találkoztak, másrészt az, hogy ugyan hallottak már a nyílt forráskódú rendszerek egyikéről-másikáról is valamit, de annyit messze nem, hogy azok jelentőségéről fogalmat alkothassanak. Az általános alulinformáltság azonban csak részben köszönhető a hegemóniára törő, zárt rendszerét terjesztő multik hathatós marketingmunkájának. A döntéshozók fogyasztói tudatosságáról, önmagukkal szembeni szakmai igényességéről azonban ugyanúgy árulkodik – arról, vajon engedik-e, hogy a PR-munka könnyű prédájává váljanak, a társdalom által döntési jogosultságokkal felruházott csoportként is, ahogyan egyenként, személyesen is.

Nem egyszerű persze a fent felsorolt döntéshozók sorsa sem, hiszen nem tartoznak ahhoz a generációhoz, amelyik

Géptermi gyakorlat
Fotó: Jankovich Oszkár

– ha nyitott volt rá – már puszta érdeklődésből is megismerkedhetett a nyílt forráskódú szoftverekkel, ugyanis gyerekkorában, otthon megvolt az erre alkalmas számítógép, és ezért nemcsak a nagyszoba könyvespolcának a kínálata jelentette számára tudásának a megalapozását, hanem a világháló is. Ez viszont azt is jelenti, hogy generációs bomba ketyeg a döntéshozók és informatika-oktatásról rendelkező döntéseik alanyai között. Példákkal illusztrálva: az informatikatanárok egyre nagyobb része kénytelen a diákjaitól tanulni egyre többet, mert elavul, a piac által redukált az informatikai tudása; a cégvezetők, a menedzserek, az intézményigazgatók, sőt a miniszterek már ma is egytől egyig teljes mértékben kiszolgáltatottak a rendszergazdáiknak. Nem nagy jósteljesítmény belátni, hogy Linux-oktatás nélkül az informatikai tudáskülönbség exponenciálisan növekedni fog.

Mindenféle kormányzati segítség nélkül, ingadozó szakmai és civil támogatással, valamint egyre megbízhatatlanabb vállalati szponzorációval a háta mögött próbál ez ellen 2002 óta tenni Rózsár Gábor[1], a Linux az oktatásban rendezvénysorozat ötletgazdája és szervezője. A számítástechnika-tanárként és rendszergazdaként dolgozó fiatalember Sallai Andrással másfél hónap alatt szervezte meg 2003-ban az első LOK[2]-ot, amelyre egyből száznál többen mentek el, közöttük 64 tanár is. Ők alkotják a LOK kemény magját, és alkotják a szervezők zömét ma is. A második LOK[3]-ra még ugyanabban az évben sor került. A következő két esztendőben újra két-két konferenciát és szemináriumsorozatot sikerült megrendezni, majd beköszöntöttek az ínséges idők évi egy-egy LOK-kal, 2008-ban pedig egyáltalán nem volt LOK. Ezért is ér fel egy reménysugárral az idei rendezvény, amelyre áprilisban kerülhetett sor. Rózsár Gábor tapasztalata az, hogy a vállalatok egyre kevésbé hajlandóak szponzorálni a hasonló rendezvényeket: „A helyzet évről-évre romlik. Idén a válság külön betett a céges támogatásoknak, és úgy látjuk, nagyon kevés cég lát üzletet egy oktatási konferencián való részvételre, még akkor is, ha a konferenciára nemcsak az oktatásban dolgozók jelentkezhetnek.”

A hét év alatt a tematika is bővült, és igyekszik figyelembe venni a különböző felkészültséggel érkező hallgatók eltérő igényeit. A LOK2009[4]-en öt szekció[5] munkájában le-

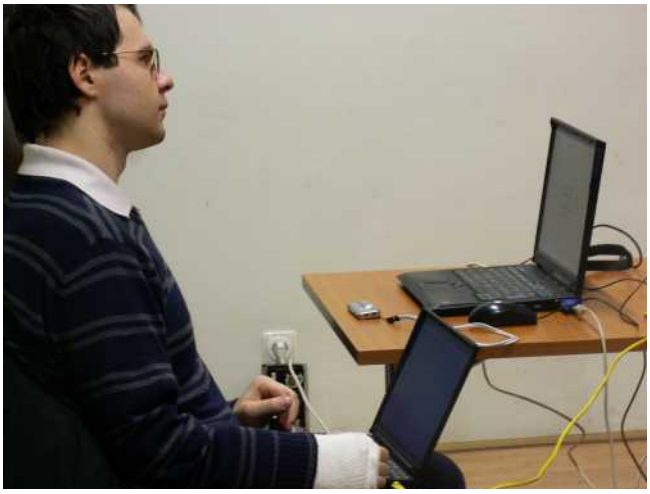
A Google Web Toolkit programozása

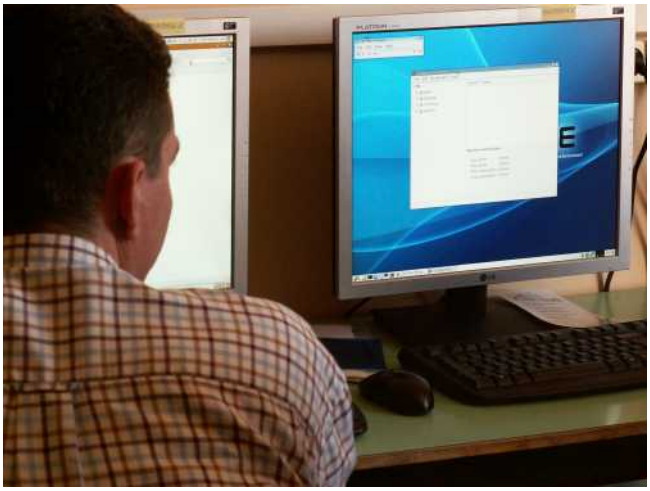
hetett részt venni. A „hagyományos” szekció leginkább az informatikai érettségi[6] vizsga során előforduló feladatok megoldásában és oktatásában segített. A főbb irányok itt a szövegszerkesztés, a táblázat- és adatbázis-kezelés, az internetes böngészők képességeinek a kihasználása, valamint a képszerkesztés voltak. Az „EDU” szekció az Elektronikus Tanulmányi Nyilvántartás rendszerének ismertetése után az ILIAS-alapú távoktatással, a Unix programozási környezettel, majd a szabad szoftveres zenei felhasználási lehetőségekkel foglalkozott. A kezdők a géptermi gyakorlatokon ismerkedhettek meg az elterjedtebb GNU/Linux disztribúciók telepítésével és használatával, míg a leginkább az iskolai rendszergazdákat célzó „Admin” szekció az intranet, a hálózatok, a csoportmunka, a hardver-választás, és a virtualizáció rejtelmeibe avatta be az érdeklődőket. A vegyes tematikájú „Szabad szoftver nap” nevű szekció pedig egyebek mellett a Google-programozás lehetőségeivel, a rendszernaplózással, és a green computing intézményi bevezetésével foglalkozott.

Rózsár Gábor szerint azért is nehéz a megfelelő tematika összeállítása, mert nincs nyilvántartás arról, hogy a magyarországi általános és középiskolák közül hányban használnak, illetve oktatnak nyílt forráskódú rendszereket. Mint mondja, a LOK által régebben elindított adatgyűjtés kudarcba fulladt, mert az iskolák önkéntes adatszolgáltatása akadozott. Kimutatás arról sincs, hogy mely hazai középiskolákban lehet Linux-rendszerek használatával érettségit tenni, ám információik azt erősítik, hogy a diákok részéről is külön erőfeszítésre van szükség, ha e törvény adta lehetőséggel élni kívánnak. Az Oktatási Minisztérium vonatkozó útmutatójából[7] is csak annyi tudható meg, hogy jelenleg összesen két, magyar fejlesztésű, nyílt rendszeren van mód az érettségire, UHU-n, illetve Sulixon. Mivel a két disztribúció soha sem tartozott a világ legelterjedtebb rendszerei közé, ezért ma már, amikor minden fontos program magyarul is elérhető, teljességgel megmagyarázhatatlan, hogy az Oktatási Minisztérium miért éppen ezeket engedélyezi és miért nem azokat a nyílt forráskódú Linux disztribúciókat, amelyekkel a diák nagy valószínűséggel a későbbi munkája során is találkozni fog (pl. Ubuntu, Debian, openSuse, Mandriva, vagy Fedora).

Az, hogy egyáltalán van-e egy iskolában valamilyen Linux

Szünetben
Fotó: Jankovich Oszkár





Haladóknak, kezdőknek
és érettségizőknek.

a gépeken, az Rózsár Gábor szerint: „leginkább a tanárokon múlik. Például hiába van ott egy lelkes rendszergazda vagy tanár, az iskola többi tanárának az ellenállásán általában elbuknak a próbálkozások. A többség nem ezt szokta meg, és idegenkedik minden újtól.” A LOK-alapító meggyőződése, hogy ebből a zsákutcából csak sok fiatal, nyitott tanár vezetheti ki az oktatást, valamint az, ha az iskolák számára a kedvezményesen adott, zárt, kereskedelmi programok állami támogatását megszüntetnék és ezzel párhuzamosan gyakori BSA-ellenőrzéseket indítanának az iskolák ellen, amelyek egyébként az ingyenes helyett sokszor a lopott rendszereket használják, akár még olyan fontos célokra is, mint az érettségiztetés.

Hogy a jelenleginél egy kicsivel több Linux legyen a hazai oktatásban, ahhoz az egyéni kezdeményezésekre épp annyira szükség van, mint a törvények és a szabályok megváltoztatására. A Linux persze már amúgy is ott van a közoktatásban, de aki gátolja a további terjedését, az ezzel a jövő nemzedék informatikai analfabetizmusát táplálja. Gutenberg[8] után a kódexmásolókat sem sikerült visszagyömöszölni a kolostorokba.

Jankovich Oszkár

Kapcsolódó hivatkozások:

[1] http://muszashi.freeweb.hu/
[2] http://www.lok.hu/
[3] http://www.lok.hu/info.html
[4] http://www.lok.hu/2009/2009.html
[5] http://www.lok.hu/2009/program01.html
[6] http://www.oh.gov.hu/main.php?folderID=3466&objectID=5007221#1
[7] http://www.oh.gov.hu/letolt/okev/doc/ketszintu_erettsegi_2009maj/infalapism_irasbeli_szoftverlista_2009maj.pdf
[8] http://hu.wikipedia.org/wiki/Gutenberg-galaxis

Tudtad-e?

Szorosabban együtt: FLOSSzine, EPA, CC, Wiki

2009. május 20. óta a FLOSSzine összes száma az Országos Széchényi Könyvtár[1] Elektronikus Periodika Adatbázis Archívumából[2] is elérhető[3]. Az EPA a Magyar Elektronikus Könyvtár kezdeményezése, amely jelenleg több mint 1300 periodikát tartalmaz, 15 tematikus csoportba rendezve. Kiadványunk a "Műszaki tudományok, gazdasági ágazatok" tárgykör alatt kapott helyet.

A MEK Egyesület május 28-i közgyűlésén egyébként a MEK és a Creative Commons Hungary[4] együttműködése is szorosabbá vált, miután mindkét részről elhangzott, hogy igyekeznek figyelemmel kísérni egymás tevékenységét, és felhasználni egymás eredményeit a közkincsek publikálásánál. Ugyancsak partnerséget ajánlott a MEK-nek a közgyűlésen a Wikimédia Magyarország Egyesület[5], amely a Wikimédia-projektcsalád[6] magyar nyelvű változatait gondozza.

[1] http://www.oszk.hu/index_hu.htm
[2] http://epa.oszk.hu/
[3] http://epa.oszk.hu/html/vgi/boritolapuj.phtml?id=01558
[4] http://www.creativecommons.hu/civi/
[5] http://wiki.media.hu/wiki/Kezd%C5%91lap
[6] http://wiki.media.hu/wiki/A_Wikim%C3%A9dia-projektcsal%C3%A1d

ár alatt, ár felett

Közkínlódás

Mit jelent, és mit nem a „szabad szoftveres” közbeszerzési kiírás

Első nekifutásra egyszerűen azt a kiírást jelenti, amelyet ez év áprilisában jelentett meg[1] a Központi Szolgáltatási Főigazgatóság[2] azzal a céllal, hogy előmozdítsa a nyílt forrás és a nyílt szabványok helyzetét Magyarországon.

Nos, a kiírás valóban lehetővé teszi nyílt forrású szoftverlicencek beszerzését, akár oktatási intézmények számára is, de nem definiálja, mit is ért nyílt forrás alatt. Valóban komoly összegeket fordít a nyílt szabványok révén megvalósítandó együttműködésre, de ezen szabványokról sem mond semmi többet. Így hát egyelőre nehezen belátható, mire és mire nem jut ebből a keretből. Homályban marad, hogy vajon a nyílt forrás egyszersmind szabad szoftvereket is jelent-e, de akkor miért vásárolnánk őket, hiszen korlátozás nélkül hozzáférhetőek? Vagy hogy egy webes alkalmazás is beszerezhető-e a keret terhére, hiszen az is megvalósíthat együttműködést olyan nyílt szabványokon keresztül, mint a http, a html, vagy a css? Szóval kicsit sárga, kicsit savanyú, de a miénk. – Ne legyünk azonban telhetetlenek: mindenképpen értékelendő, hogy ez a kiírás egyáltalában megtörténhetett. Lássuk, miként látja az előzményeket és a lehetséges következményeket **Baja Ferenc**, a Miniszerelnöki Hivatal[3] politikai államtitkára, informatikai kormánybiztos, **Balsai Péter**, a Szabad Szoftver Intézet[4] ügyvezetője, valamint **Dr. Szentiványi Gábor**, a Linux Ipari Szövetség[5] elnöke, egyben az ULX Kft. ügyvezetője.

FLOSSzine: *Mit gondol, nem támadják-e meg ismét a „vagy azzal egyenértékű” kifejezés miatt a kiírást, hiszen az előző eset miatt a GVH[6] jelenleg is perben áll?*

B. F.: Szerintem nem, én azt gondolom, hogy már megtették volna. (…) Nyilván a „szabad szoftveresek” nem fogják kifogásolni, mert ez az ő számukra kedvező. Megtámadhatják más típusú platformok, de nincs tudomásom róla, hogy ezt terveznék jelen pillanatban, azonban nyilvánvalóan nem zárom ki. Ez esetben a bíróság eldönti, hogy mi történjen. Szándékaink ettől függetlenek.

Sz. G.: A kiírást minden évben újra ki kell írni, ez szinte egy kényszer, mivel ha ez nem történik meg, akkor nem lehet központosított beszerzéssel a keretből lehívni. Felvetül a kérdés, hogy ha nincs keret, akkor mi történik. Akkor egyedi megrendelések vannak, de nem a keret terhére. Akkor nincsenek fixálva az árak, amiket így fel lehet verni, ami így még drágább is lehet, mint kerettel. A keret eredeti célja az, hogy csökkentse az árakat. Itt nyilván nem a kerettel van baj, hanem annak tartalmával. Mindegy, hogy mit írnak oda, hogy Microsoft, vagy hogy Novell, vagy IBM, vagy RedHat, vagy bármi, mivel szinte soha nincs korrekt indikációja a konkrét gyártó megnevezésének. A „vagy azzal egyenértékű” kifejezés beleírása egy EU-s kötelezettség, amivel lehet élni és visszaélni. Sajnos sokszor az utóbbi történik. Mindent meg lehet támadni, azonban meg kell nézni, hogy a kiírás valóban előre mutat-e az eddigi helyzethez képest, és ennek szellemében kell cselekedni.

B. P.: Szerintem ez egy nagyon balszerencsés, a célt eltévesztő kiírás, mely támadható, s talán hasznos is ha támadások érik, mert az ad arra esélyt, hogy a jövőre nézve valami érdemi szülessen. Azt hiszem, ha komolyan veszik, a következő szöveg miatt vagy értelmezhetetlen, vagy teljesíthetetlen (ti. a kiírás – a szerző): „nyílt szabványokon keresztüli teljes együttműködést biztosító közigazgatási szoftverlicencek bővítésére, kiegészítésére, meghosszabbítására, verzió-követésére, cseréjére, valamint új szoftverlicencek beszerzésére és kapcsolódó szolgáltatások teljesítésére”. Mi az, hogy teljes együttműködés? Mi az, hogy nyílt szabvány? Mert csak ennek a két válasznak az ismeretében lehet azt megmondani, hogy valami „nyílt szabványokon keresztüli teljes együttműködést” biztosít-e? (…) A szöveg alapján értelmezhető úgy, hogy négy év alatt vegyünk 12 milliárdért Microsoft szoftvert és hozzá tartozó szolgáltatást, 6 milliárdért Novell által forgalmazott szoftvert és hozzátartozó szolgáltatást, 6 milliárdért meg bármit. A Novell (3., 4. rész) és a bármit (5., 6. rész) blokkba beférhetnek esetleg mások, Redhat, Multiráció, IBM, stb. is. Nos ez jó, mert minél több cég fér bele, annál inkább nehéz lesz belekötni az eredménybe. Szerintem nem látszik a kiírásból az a cél, melyet mi reméltünk a nyilatkozatok alapján látni, hogy végre olcsóbb, részeiben lecserélhető (emiatt versenyt támasztó) állami informatika kialakítása a cél, mely szabad és nyílt forrású szoftvereken alapszik.

FLOSSzine: *Jelent(het)-e ez a kiírás egyszersmind szemléletváltást a megrendelők, a közoktatás, a közintézmények oldalán?*

Sz.G.: Ezért minden szereplőnek tennie kell. Annak is, aki ebből vevőként profitálni akar, és szállítói oldalon is. Társadalmi, politikai értelemben is tenni kell érte. Sajnos ez nem megy magától, ez csak egy lépés a jó irányba. (…) Magyarországon szabad szoftverek általános beágyazottsága nem mély, az egész gondolati kör még hiányzik. Először is el kell juttatni ennek az üzenetét. Addig viszont hiába juttatjuk el, amíg nincsen meg ennek a támogatói a háttere kormányzati szinten. Most talán ennek a csírái nőnek ki, ami a tehetetlenség ördögi körét átvághatja.

B.P.: A szemléletváltásnak meg kell előznie a pénzköltést, mert ha a jól dolgozunk, akkor ugye arra a célra költjük a pénzt, amit el akarunk érni. Márpedig, ha nem történt meg a szemléletváltás, akkor nem fog nagyon másképpen és másra sem történni a pénzköltés. Ez a kiírás szerintem igazolja ezt. Hiszen nem történt szemléletváltás és elsődlegesen Microsoftot és Novellt akarunk venni. Az állami és közigazgatási informatika, valamint az arról való gondolkodás is egy megmerevedett rendszerben történik. Ez a merevség ott van az oktatásban és gyakorlatilag újra termeli saját magát. Az, hogy nem ismerjük fel a változás lehetőségét, árt a saját gazdaságunknak, munkahelyeket vesztünk miatta, és egyre közelebb kerülünk ahhoz, hogy elsődlegesen más gazdaságoknak bevételt termelő informatikai piaccá váljunk.

FLOSSzine: *Érkezett-e felkérés az állami szervezetek részéről az ODFA[7], SzSzI, vagy a LIPSZ önhöz/feléd más csatornán törvény-, döntés-előkészítési feladatokban történő részvételre?*

Sz.G.: Itt proaktívnak kell lenni, tehát nekünk kell azt mondani, hogy igenis szükség van ránk. Azt nem lehet mondani a kormánynak, hogy nem csináljátok jól, azt viszont lehet, hogy ha ebben nincs tapasztalatotok, akkor mi szívesen segítünk. Azt, hogy jöjjön egy megkeresés, valószínűtlennek tartom.

B.P.: Az SzSzI és az ODFA, de a többi civil szervezet is nagyon sokszor kérés nélkül is mondja, hogy szerinte mit kell tenni. Az SzSzI másokkal együtt most fejezte be egy sok száz oldalas anyagát a MEH felkérésére (…) Az SzSzi próbál szabad szoftveres megoldásokat kifejleszteni, vagy vásárlás által elérhetővé tenni. Vegyes sikerrel. Az a tapasztalatunk, hogy egyre több területen születnek olyan szabályozások (jellemzően rendeleti szinten), melyek megdrágítják, vagy éppen lehetetlenné teszik a szabad szoftverek készítését az adott feladathoz. Annak a jogi háttere, hogy pontosan hogyan is kell a szabad szoftvereket illetve az azokon végzett fejlesztői munkát számvitel illetve adó szempontjából helyesen kezelni, tisztázatlan. Ennek rendbetétele szükséges, ezt látjuk, de jelenleg erőforrásainkat meghaladja. Talán arra lesz energiánk, hogy a problémát jelezzük, a köztudatba bejuttassuk.

FLOSSzine: *Szükséges-e, érdemes-e bevonni a nyílt forrású és szabad szoftverek, műszaki szabványok terén jártas civil szervezeteket, kutató/oktatási intézményeket, hazai és külföldi vállalkozásokat?*

B.F.: Mikor bejelentettük ezt egy sajtótájékoztatón, akkor az IVSZ[8] (Informatikai Vállalkozások Szövetsége – a szerző) elnöke is jelen volt az én kérésemre azért, hogy érzékeltessük, hogy az állam ezt egyedül nem fogja tudni megoldani, tehát próbáljuk behívni azokat a magyarországi partnereket, akiknek egy üzleti szeletet tudunk ezen keresztül megjeleníteni és azt tudjuk mondani, hogy látunk egy olyan piaci rést, ahova – pláne a gazdasági válság keretei között – be tudnak lépni. A nagyobb kérdés itt az lesz, hogy ezek a szerveződések sokféleségükben – ami egy pozitívum – képesek-e együtt egy erős szövetségi konglomerátumot létrehozni a hagyományos szoftvergyártók és forgalmazók alternatívájaként, hiszen a sokszínűségből is muszáj egy egységet létrehozni, amikor piacra lépnek. Hogy ezt létre tudják-e hozni, vagy sem, azt nem tudom. Mi készen állunk arra, hogy ha ez elakad, segítsünk, de ez az ő feladatuk, felelősségük.

Sz.G.: Magyarországon azt szokás mondani, hogy gyenge ezen a területen sokszor a civil társadalom. Szerintem is kicsit gyenge, mindenki magának sütögeti a pecsenyéjét és komoly lépést egyedül nem tud megtenni. Vannak nemes eszmék – immateriális eszmék – és nagyon könnyen mellékjük lehetne állni, csak azért, mert ha az az eszme beépül a társadalomba, akkor abból mindenki profitál. (…) Nő azonban a civilek jelenléte, és ezt a kormány is észrevette. Ami a multikat illeti, én azt látom, hogy Brüsszelben – ahová rendszeresen járok ki mint tanácsadó – együtt harcolnak olyanok, akik ősellenségei egymásnak piacilag, azonban közösen össze tudnak rakni egy álláspontot, ami Magyarországon nem így van. Nincs olyan eszme, vagy közös irány, ami mellé odaállna több vállalat teljes mellszélességgel.

B.P.: (…) Szerintem a cél megfogalmazásában kellene együttműködni civil és nem civil szervezeteknek, majd a lebonyolítás tervezésében szakmai szervezeteknek, de erős kontroll mellett, mert az informatika területén a szakmai szervezetekben a klasszikus nagyok dominálnak, akiknek lényegüknél fogva az a céljuk, hogy minél nagyobb bevételt realizáljanak, s nekik a cél (a bevétel) eléréséhez mind pénzük, mind egyéb eszközeik vannak. Ez a dolog nem feltétlenül az ördögtől való, egyszerűen csak így van. Az is

fontos, hogy nekünk itt és most nem másokat utánozva, hanem magunknak megfelelően a cél elérése érdekében kell jól dolgoznunk. Nekem a külföldi tapasztalatok magyar adaptálása kapcsán valamiért mindig Rejtő Láthatatlan légiójából egy rész[9] jut az eszembe.

FLOSSzine: *Miként látja/látod a szabad szoftverek és a nyílt forrás jelenét és közeljövőjét Magyarországon?*

Sz.G.: Nem hiszem, hogy egyik pillanatról a másikra áttörés lenne. Ezért minden szereplőnek tenni kell. Annak is, aki ebből vevő oldalon profitálni akar, de szállítói oldalon is, társadalmi, politikai értelemben is tenni kell érte, sajnos ez nem megy magától. (…) [Ez másodszor volt!] Amíg például a tiszta szoftver program ilyen mértékig korlátos, nem kiegyensúlyozott, addig az a helyzet, hogy a keret terhére az egyik esetben egy minisztérium bevásárol évi 1,7 milliárd forint erejéig, a másik esetben meg nem. Ez nyilván tarthatatlan a versenysemlegesség szempontjából is, a társadalom számára pedig nagyon cinikus üzenet.

B.P.: Ez a kiírás változásokat okoz, hiszen megbékéltet. Érdemi változás eléréséhez szerintem egy-másfél éves előkészítés, több éves munka kell. Vicces, de egy kormányzati ciklus, ami ugye 4 év, valójában (2 és fél év érdemi munka) nem is lehet elég rá. Szóval több, a téma iránt érdeklődő, a célt és a tervet felvállaló politikus kell ahhoz, hogy valami valóban változzon. Az állam és közigazgatás csak a szabályozás változása után változtatható meg valójában. Itt óriási a tehetetlenségi erő, tehát ez egy nagyon nehezen és lassan változtatható terület.

Pfeiffer Szilárd

[1] http://www.kozbeszerzes.hu/lid/ertesito/pid/0/ertesitoHirdetmenyProperties?objectID=Hirdetmeny.portal_6918_2009
[2] http://www.kszf.hu/
[3] http://www.meh.hu/
[4] http://www.lipsz.hu/
[5] http://www.szszi.hu/
[6] http://www.gvh.hu/gvh/alpha?null&m5_doc=5391&pg=72
[7] http://www.odfalliance.hu/hu/index.html
[8] http://www.ivsz.hu/
[9] http://mek.oszk.hu/01000/01034/01034.htm#7

További információk:

http://www.openskm.com/sajto/index.html
http://computerworld.hu/ballmer-budapesten.html
http://computerworld.hu/25-milliardos-panasz.html
http://computerworld.hu/microsoft-szoftverek-ujabb-fordulo.html
http://net.jogtar.hu/jr/gen/hjegy_doc.cgi?docid=A0300129.TV&
http://www.gvh.hu/gvh/alpha?null&m5_doc=5391&pg=72
http://www.sg.hu/cikkek/62367/a_gvh_magyarazata_a_microsoft_perr_337_l
http://tasz.hu/kozerdeku-lobbi?page=2

Tudtad-e?

Minden ötödik legális szoftver nyílt forrású vagy szabad!

A BSA (Business Software Alliance) 110 országra kiterjedő éves tanulmánya[1] szerint - amely már a hatodik a sorban - 2008-ban az illegális szoftverhasználat mértéke világszerte elérte a 41 százalékot. A vizsgált szoftverek közül a szabad és nyílt forrásúak aránya eléri a 15 százalékot. Csak a törvényes használatot figyelembe véve az arány egy a négyhez, azaz a legálisan alkalmazott szoftverek 20 százaléka nyílt vagy szabad forrású. A tanulmány régiókra és országokra bontva is ismerteti az illegális szoftverhasználat mértékét, illetve annak változását az előző évekhez képest. Hazánk, a maga - évek óta stabilnak mondható - 42 százalékával épp csak lemarad a legalacsonyabb arányt felmutató országok előkelő listájáról. Összehasonlításképpen: a közép-kelet-európai térségben 66 százalékra, míg az EU országaiban 35 százalékra taksálja a törvénytelen szoftverfelhasználás átlagát a világ több mint 80 országában tevékenykedő szervezet. A sereghajtó Grúzia, ahol 100 szoftverből mindössze öt legális és még a sort vezető USA-ban is minden ötödik szoftver illegális. Számot-

tevő változás az illegális szoftverhasználat mértékében az előző év 38 százalékához képest nem állt be, ugyanakkor a 2004-2006-os időszak stagnálása (35 százalék) most mintha emelkedő tendenciába fordulna.

A szabad szoftverek számos esetben segíthetnek az illegális használat mértékének csökkentésében, hiszen korlátozás nélkül, rendszerint ingyenesen hozzáférhetőek. A vállalati szféra mellett erről a lehetőségről a lakosság sem kellően tájékozott, annak ellenére, hogy ebben a körben igazán gyakori a szoftverek jogosulatlan használata, pedig a funkciók könnyen kiválthatóak lennének szabad szoftverekkel. Egy otthoni felhasználó kívánalmainak, amelyek a legtöbb esetben nem mutatnak túl egy böngésző, egy levelező és egy irodai programcsomag használatán, számos, magyar nyelven is elérhető Linux disztribúció eleget tud tenni. Erről a tényről azonban említést sem tesz a Tiszta Szoftver kampány[2], amelynek financiális hátterét az állami költségvetés adja. A BSA-kampány és a hozzá kapcsolódó program jelenleg az adóforintokból csupán szoftverlicencek vásárlásának a támogatását teszi lehetővé, terméktámogatás (telepítés, karbantartás, oktatás) vásárlását azonban nem. Ezzel pedig egyenesen távol tartja a lehetséges érdeklődőket a szabad szoftverek használatától, ahelyett, hogy valós segítséget adna nekik.

[1] http://global.bsa.org/globalpiracy2008/index.html
[2] http://tisztaszoftver.hu/

Tudtad-e?

Ellopták a lopásgátlót?

A legérzékenyebb pontján ejtettek sebet a világ vezető szoftvergyártóján. A Microsoftot április közepén ugyanis egy olyan védett szabadalom jogtalan felhasználása miatt ítélte 388 millió dollár (kb. 77 és fél milliárd forint) kártérítésre[1] egy amerikai szövetségi bíróság első fokon[2], amelyikre egyes termékeinek az eladását alapozta. A szoftveróriást hat éve perlő – eredetileg ausztrál, de ma már amerikai – 40 fős Uniloc USA Inc. nevű cég[3] szabadalmaztatta azt a technológiát, amelyet sok alkalmazásfejlesztő vállalat használ úgynevezett shareware[4] termékeinek végeladási licenceléséhez. Ez a Uniloc „fizikai eszközfelismerő” technológiája, amely a játékprogramokban "SoftAnchor" néven található meg, de ugyanennek a biztonsági eljárásnak a „NetAnchor” nevű változatát használja többek között egy multinacionális olajtársaság is, amelyik számítógépeinek a fizikai alapú hitelesítésével védi hálózatát az illetéktelen behatolóktól[5].

A licencvédő program a letöltött, zárt forráskódú alkalmazások részeként képes a célgép hardvereszközeit beazonosítani azok egyedi, belső, fizikai jellemzői alapján, majd létrehoz belőlük egy digitális ujjlenyomatot[6]. Ezzel az ujjlenyomattal regisztráltathatja a szoftvert az eladónál a vásárló, amikor – az időkorlátos kipróbálási lehetőség lejárta előtt – online átutalással kifizeti a terméket. A Microsoft ilyen technológiával terjeszti évek óta például a Windows XP-t és a Microsoft Office XP-t. Az óriáscég fellebbezett az ítélet ellen[7], mondván, hogy egyrészt szerinte érvénytelen a kérdéses szabadalmi bejegyzés, továbbá, hogy más, saját fejlesztésű technológiát használ lopásvédelemre.

[1] http://uniloc.web4.hubspot.com/press-releases-0/bid/20863/Uniloc-Awarded-388-Million-in-Damages-in-Major-Patent-Infringement-Case-Against-Microsoft
[2] http://www.iptrademarkattorney.com/2009/04/patent-attorney-jury-award-388-million-microsoft-uniloc-patent-infringe.html
[3] http://www.uniloc.com/
[4] http://en.wikipedia.org/wiki/Shareware
[5] http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9132801
[6] http://uniloc.web4.hubspot.com/press-releases-0/bid/20863/Uniloc-Awarded-388-Million-in-Damages-in-Major-Patent-Infringement-Case-Against-Microsoft
[7] http://www.iptrademarkattorney.com/2009/04/patent-attorney-jury-award-388-million-microsoft-uniloc-patent-infringe.html

Háború, Linuxon

Bos Wars

A játékrovat sokszínűségének egyik ellensége az, ahogyan a versenyképes játékprogramok szabad rendszeren leginkább a belső nézetű lövöldözős kategóriából kerülnek ki. Mégsem reménytelen feladat halványítani e furcsa tényt, hiszen nem ritkán születnek értékes szimulációs, ügyességi vagy éppen stratégiai játékok is.

Nos, a bevezető rész mondandóját mégse lépjük át ilyen könnyedén, inkább elemezzük egy kicsit! Talán ennyire népszerű az „én ölök” stílus? Esetleg a Linux felhasználói ilyen mértékben függenek az FPS kategóriától? Természetesen nem. Mindössze arról van szó, hogy a piacvezető lövöldözős játékok már gyárilag Linux-képesek. Az igazán nagy klasszikusok pedig eleve „moddolhatóak”, ami által a játékmotor számunkra érdekes verziója tucatnyi egyéb linuxos projektet hajthat meg élete derekán. Sőt, a fő szoftverek kifutó alapkódja sokszor GPL licenc alá sorolva segíti egy újabb, többplatformos (ám kategóriájában hasonló) remekmű megszületését.

Más műfajra tekintve azonban már közel sem ilyen rózsás a helyzet: RTS vonalon például a legjobb szándékkal sem tudok megnevezni olyan fejlesztő csapatot, mint a saját kategóriájában vezető id Software. Hiszen hiába a Blizzard nagysága, a Warcraft sorozat epizódjai nem támogatják a Linuxot. Ránk nézve még a C&C 3: Tiberium Wars is érdektelen, mivel ennek a fejlesztői sem rajonganak a pingvinért. A költői kérdés ezután szinte adja magát: mi marad nekünk a népszerű stílusból? Joggal számíthatunk a külsős porterek munkájára (HomeWorld), vagy épp egy teljesen idegen alapkódra húzhatjuk rá a kiválasztott stratégiai játék világát (Warcraft 2). Esetleg szóba jöhet még a Win32 wrapperek használata (Starcraft, Warcraft 3), de a teljesen szabadon elérhető, jellemzően multiplatformos képességű projektekből is szemezgethetünk (Glest). Ezzel a kategóriával pedig el is jutottunk a Bos Wars-ig: a címben említett program szabad forráskóddal rendelkezik - minden ereje és esetleges elmaradása innen ered.

Íme a Bos Wars, KDE asztalon.

Minden kezdet nehéz...

A szerény felderítő csapat maradványa.

Múlt, jelen, jövő

Viszonylag öreg kezdeményezésről van szó: a Bos Wars már évekkel ezelőtt is létezett (Battle of Survival néven). A játék alapjául a sokszor bizonyított Stratagus motor szolgál, amire a készítő csapat valósággal „rávarrta” a saját grafikáját és szabályait. A kiforrott alapkód 2007 júniusa óta nem frissült, két okból kifolyólag: az utolsó kiadások funkcionalitása teljes, valamint a fejlesztők mára már kifejezetten csak a Bos Wars karbantartásán dolgoznak. Maga a játék megújulás-párti: az időnként felbukkanó friss verziók új egységekben és pályákban, valamint finomított grafikában térnek el az elődöktől. Száz szóval sem tudom kellően nyomatékosítani: a motor és a projekt remekül összeillenek, így a közeli jövőre nézve senkinek se legyenek kétségei az életképes kiadások felől. Érdemes tehát aktívan használni a játék honlapját, és sűrűn olvasgatni az ide vonatkozó híreket.

Na de mégis, mi ez?

Ha nagyon egyszerűen szeretném definiálni a Bos Wars-t, akkor egy szabad forrású Starcraft klónnak nevezném. Valójában persze nem erről van szó. A dokumentációk szerint egy letisztult, valós idejű stratégiai játékról beszélünk, ami egy elképzelt jövőbeli háború kellős közepébe repíti a felhasználót. A programnak a cikk írásakor sajnos még nincs kerettörténete, valamint az (összefüggő) egyjátékos Campaign módja is hiányos. Viszont már most több tucat egységgel, és legalább ennyi pályával várja a hálózati (illetve a számítógép ellen viselt) háború vezéreit. Grafikai kivitelezése a néhai Warcraft 2 szintjén áll, a híres-neves Starcraft izometrikus 2D képi világát alulról súrolva. Játékmenetét vizsgálva sem kell különösebben szaporítanom a szót: a lehető legrövidebb időn belül ütőképes haderőt kell építeni az ismerős elgondolások alapján. Kristálymezők, titánium lelőhelyek biztosítják a szükséges alapanyagot, az építendő generátorok és reaktorok pedig az energiát. Gyalogos és gépesített egységek tömkelege egyaránt bevethető a harctéren, mint ahogy az egyéb technikai lehetőségek is hasonlóan „színesek”: kamerák, radarok, automata ágyúk éppúgy rendelkezésre állnak, mint a sebesülteket gyógyító kórházak. Természetesen a diplomácia sem hiányozhat a repertoárból, így az idegen haderők három csapatba sorolhatóak: szövetségesek, ellenségek és semleges csapatok. Helyi hálózati játékra gondolva talán nem kell mondanom, mekkora buli két idegen hadtest összevonásával lerohanni a számítógép seregét!

Kipróbálnám...

Semmi akadálya, több telepítési mód kínálkozik: a projekt felépíthető forráskódból, de elérhető „kész” formában is. Azt a véleményemet, miszerint a legritkább esetben javaslom az előre fordított verziók használatát - nem szoktam véka alá rejteni, a Bos Wars viszont a ritka kivételek egyike: a gyári anyag tökéletesen használható.

A http://www.boswars.org honlapról indulva le kell tölteni a natív, linuxos binárist. Ha ez megtörtént, a személyes mappánk egyik alkönyvtárába ki kell csomagolni a lehívott archív tartalmát, majd az ott található boswars (ELF) állomány segítségével felhasználóként indítható a program. Forráskód használata esetén a kicsomagolt forrásmappában kiadott ./autogen.sh, ./configure, make depend, make parancsok vezetnek eredményre - esetleg a scons fordító is munkára fogható (persze csak akkor, ha a tekintélyes függési lista teljesíthető).

Végül íme, a gépigény: néhány száz MHz órajelű Pentium3 CPU & 128 MByte központi tár esetén már vígan át lehet kapcsolni akár nagyobb felbontásokba is. Fontos tudnivaló, hogy a személyes beállítások, valamint a mentett

Egy légi támogatású rohamosztag.

Ez a támadás sajnos elhalt.

játékállások minden esetben a /home/$/.boswars úton tárolódnak, tehát erre a rejtett könyvtárra illik nagyon vigyázni...

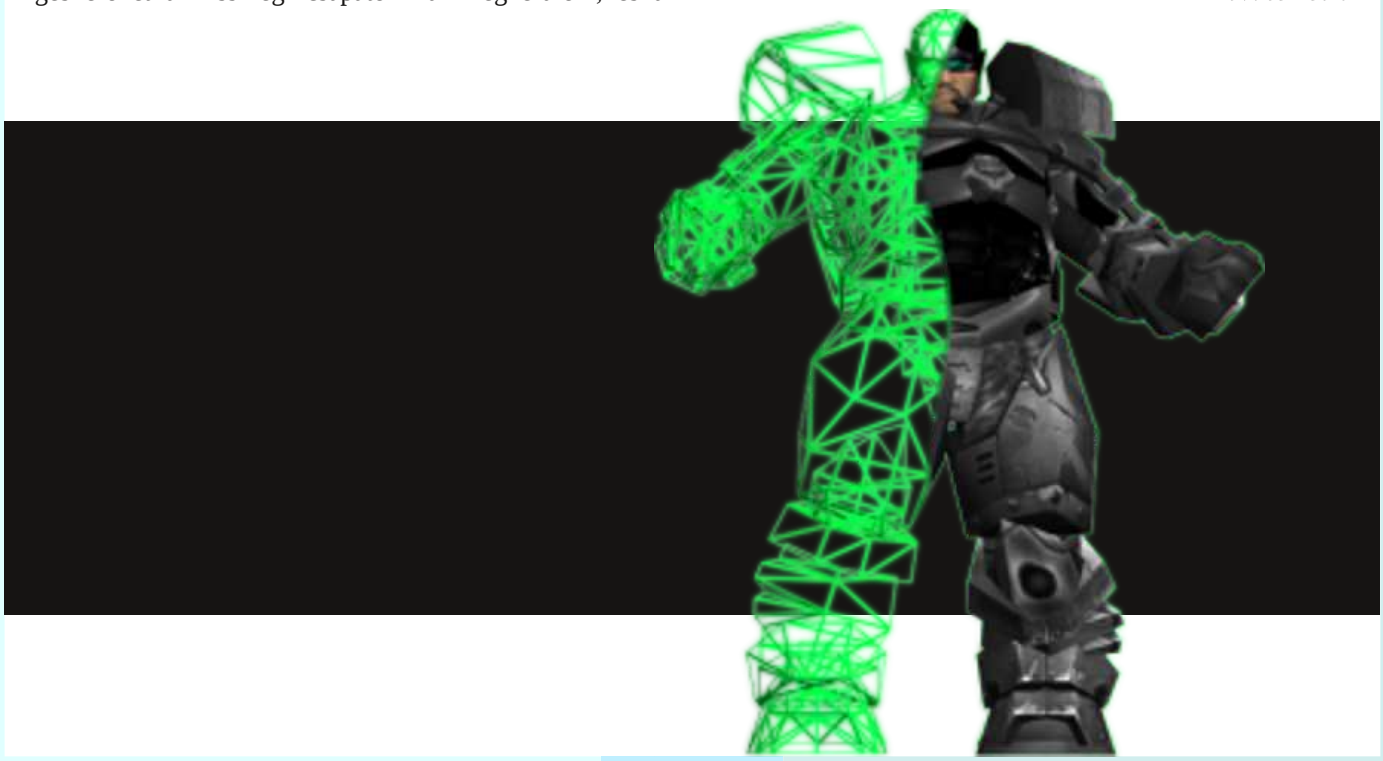
Tippek, trükkök

Mint azt írtam, talán szükségtelen bővebben érintenem a játék menetét. Ehelyett inkább néhány tippet adnék azoknak, akik ismerik a stílust, de konkrétan ezzel a programmal még nem találkoztak. Íme, a saját tapasztalataim: a számítógép jellemzően két-három ponton „szeret” támadni. Ezek közül az egyik pozíciót komoly túlerő tarja majd nyomás alatt. Ennek megfelelően megelőzöm a bajt, és szinte mindig úgy kezdek, hogy (tömör bázisra törekedve) a titánium és kristály mezők közé építem a főépületet. Az építés befejeztével a „munkás-mérnököket” azonnal bányászni küldöm, miközben közel tíz további munkást hívok életre. Az első néhányat szintén bányákba osztom, a megmaradt három-néggyel pedig rohamtempóban kezdek építkezni. Azonnal felhúzok egy kamerát és egy radart a bázis közepére, hiszen így nem érhet meglepetés támadás előtt. A kamera rádiuszára alapozva megítélem a terepet, és a leglágyabb pontokon automata ágyúkat telepítek. Gyorsan leteszek néhány generátort, eközben a gyalogsági erők felállítását is elkezdem, a légierő alapjával együtt (egyelőre felderítési céllal). Ekkorra talán már megtörténik az első ellenséges roham, amit a gépágyúk elfojtanak. Azonnal a megtört támadás irányába indulok a repülőkkel, mögöttük szorosan a gyalogosokkal: lesz ami lesz, jutok, ameddig jutok... Persze fontos, hogy a termelés továbbra is zökkenőmentesen folytatódjon. Szinte biztos, hogy a kiküldött csapatom perceken belül elvérzik. Ekkor a radar képét figyelve, az újra támadó ellenséges erőket a friss légi csapatommal megkerülöm, és az ellenfél fő bázisát bombázókkal és chopper-ekkel ízekre szedem. Elsődleges célpont természetesen a főépület, majd az összes munkás. Ha sikerrel jártam, a másik fél totálisan lebénul, hiszen képtelenné válik mindenféle utánpótlásra. A folyamatosan hizlalt földi egységeim a gépágyúkkal együtt pedig elég erősek ahhoz, hogy a bázisomnál lézengő (és utánpótlást nem kapó) támadó csapatokat megfogják arra pár percre, amíg a légierőt visszahívom segíteni. Persze ez csak az én receptem, ami könnyen fokozható például tankokkal is...

A betöltött szerep

Természetesen nem célom összevetni ezt a príma játékot semelyik naprakész, kereskedelmi RTS projekttel sem. Ha mégis megtenném, valószínűleg alulmúlná a divatos megoldásokat. Egy dologban azonban összemérhető a nagyokkal - mégpedig a költséghatékonyságban. Tehát ha most felállnék a gépem elől, és elindulnék programot vásárolni, akkor a nekem tetsző stratégiai gyöngyszemért (a cikk írásakor) nagyjából tizenegy kék hasú bankóval kellene fizetnem. Aztán miután hazahoznám a telepítő DVD-t, rögtön lentebb hagyna a jókedvem. Igen, a favorit darab jelenleg sehogyan sem fut Linuxon. A Bos Wars viszont működik, és nem kerül semmibe. Ugyan nem olyan szép, mint a fizetős társai, de ötletekben nem szűkölködik, ráadásul megelégszik egy hat-hét éves PC kapacitásával is. A grafikai téren említett „hiányérzetet” egyébként meggyőződésem szerint néhány hónap múlva újra kell majd gondolnom: nem tartom kizártnak, hogy a cikk megjelenésének idején már új küllemű generált világgal fog szembesülni a potenciális játékos.

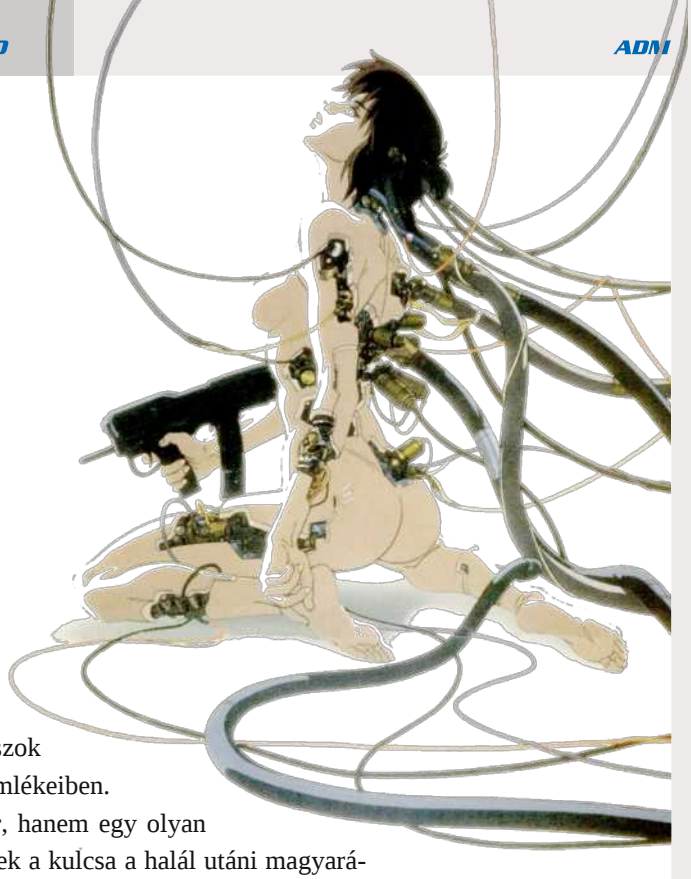
Kovács Zsolt



szoftverlélek

Szellem a gépben

...OS vándorlás heartbeat, drbd, live migration



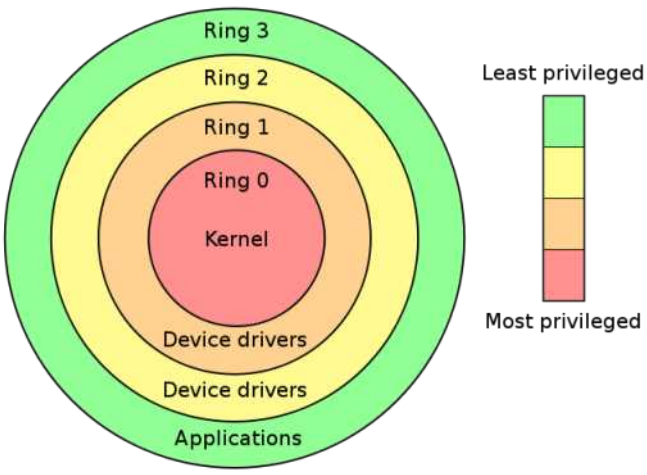
A filozófia

Az egyik örök kérdés, hogy van-e élet a halál után, gyakorlatilag egyidős az emberiséggel. A kezdeti elképzeléseket nem minden esetben ismerjük, azonban az időtállóbb válaszok napjainkban is jelen vannak a nagyobb világvallások írásos emlékeiben. Bizonyos elképzelések szerint az ember nem csak hús és vér, hanem egy olyan különleges egzisztencia, mely rendelkezik „lélekkel”. Ez a lélek a kulcsa a halál utáni magyarázatoknak. Mindenki ismeri a nyugati elképzelés, mely szerint bizonyos szűrés után, a lélek egy tökéletes, tiszta világba kerül, bizonyos „paradicsomba”. A keleti elképzelések szerint a lélek a test halála után egy másik testet ölt, s éli tovább örök / nem örök körforgását.

Informatikai szempontból az utóbbi a hasznos számunkra. Ha a testet a HW-el, a lelket pedig az SW-el azonosítjuk, akkor igencsak kívánatos dolog hogy a HW meghibásodása esetén a SW tovább „éljen”, és szolgáltasson. Cikkemben erre kínálok egy megoldást. A felhasznált eszközök külön-külön is hasznosak, és használtak, így lényegében csak egy megoldási javaslatot vázolok, melytől kísérletező szellemű olvasóim eltérhetnek, sőt javaslom, hogy térjenek el saját céljaik, igényeik irányába.

Az elmélet

A felhasznált *opensource* eszközök közül kiemelném a XEN virtualizációs technikát. A kezdetek kezdetén biztonsági szinteket határoztak meg, melyek egy-egy program jogosultságát, futási, és cselekvési szintjét jelentették. Később a portolhatóság miatt egyszerűsödött a modell, még ha a processzor támogatott is több gyűrűt, és többnyire három szintet különböztettek meg egymástól, ring0 a kernel kódnak, ring2 a privilegizált kódoknak, és ring3 a nem privilegizált kódoknak. A XEN-es virtualizáció esetében több biztonsági szintet használ a hypervisor, és ezt még kiegészítik az AMD-V és a VT-x technológiák egyéb hardveres szolgáltatásokkal. Jellemzően a XEN-nél a 0-s gyűrűben fut az úgynevezett xen-hypervisior, és ő rendelkezik a legtöbb/összes HW erőforrással. A kiemelt jelentőségű Dom0 is csak rajta keresztül érheti el a gép erőforrásait. A Dom0 azért kiemelt jelentőségű, mert ebből a xVM-ből tudjuk irányítani a többi *guest* xVM-et. Mielőtt tovább haladnánk meg kell különböztetni paravirtualizált PV, és hardver virtualizált HVM esetet. Az utóbbi esetben az összes kernel szintű hívás a xen-hypervisor-on halad keresztül, így ennek teljesítménye kisebb, azonban bármilyen OS futtatható rajta, anélkül, hogy a futtatott rendszer „tudatában” lenne annak, hogy csupán virtuális. A PV eset (ezt használják produkciós környezetben inkább) gyorsabb, mivel a *guest* rendszer módosított kernellel fut, hogy bizonyos rendszerhívásokat az ABI-n (Application Binary Interface) keresztül intézzen és ne közvetlenül az architek-



túrát célozza meg vele, gyakorlatban ez xen-patchelt kernelt jelent. HVM-et csak valamilyen virtualizációs technológiát támogató processzorral lehet megvalósítani, ha ezzel nem rendelkezünk akkor marad a PV, ami gyakran nem is baj. Persze érdemes kernelprogramozónak lenni ha teljes egészében érteni szeretné valaki a XEN működését.

A másik fontos lépés feladatunk megvalósításában, a *live-migration*. Első lépésként szükségünk lesz egy közös tárhelyre, mely többféleképpen is megoldható. Másik, sokkal bonyolultabb lépés a memóriatartalom migrálása. Ez a szűk keresztmetszet a migrációs idő csökkentésében. A memória átmigrálása három fázisban valósulhat meg.

'Push' fázis:

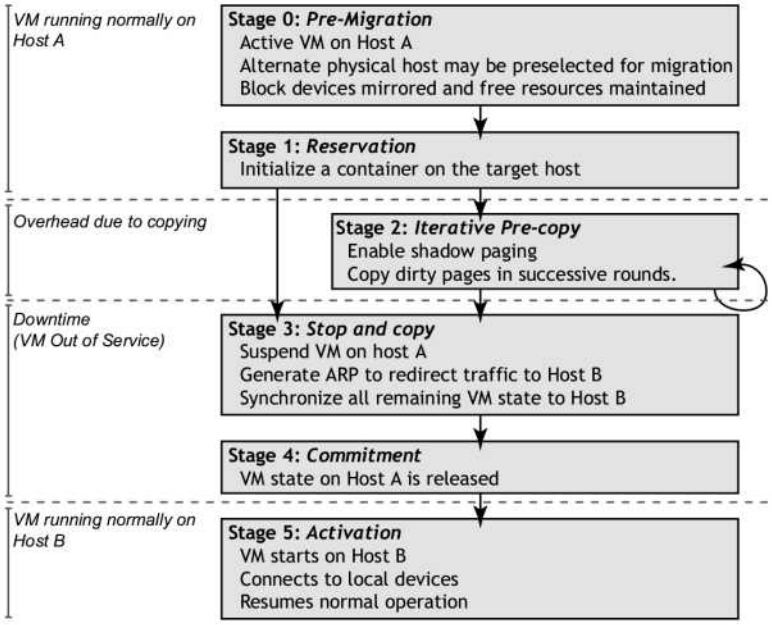
A forrás xVM fut tovább, mialatt bizonyos memória lapokat átmásol a cél xVM-re. Az adatkonzisztencia biztosítása végett a módosított tartalom újraküldésre kerül.

'Stop-and-copy' fázis:

A forrás xVM megáll, a még nem migrált memória rész is másolásra kerül. Valamint ezzel egy időben elindul a cél xVM.

'Pull' fázis:

A cél xVM elindulása után azokat a hivatkozásokat, melyek még nem kerültek másolásra a forrás gépről „magukra húzzák”.



A végeredmény egy közel „halhatatlan” operációs rendszer, mely képes a hálózat kesze-kusza ösvényein egyik gépről a másikra vándorolni, anélkül, hogy jelentősebb szolgáltatáskiesést tapasztalnánk. Azért mondom, hogy *közel* halhatatlan, mert sajnos „az ellen nem véd” ha a gép hirtelen hal el. Ugyanis ebben az esetben a memória nem tud átvándorolni, így nem tud live-migrálni. A migráció megtörténik ugyan, de ebben az esetben több másodperces szolgáltatáskiesés figyelhető meg, valamint az aktív session-ök, és egyéb nem mentett szolgáltatások (még memóriában lévő) elvesznek. Az *on-the-fly* memória szinkronizációt még nem létezik, így a hirtelen halál elleni védelemre még várnunk kell.

A gyakorlat

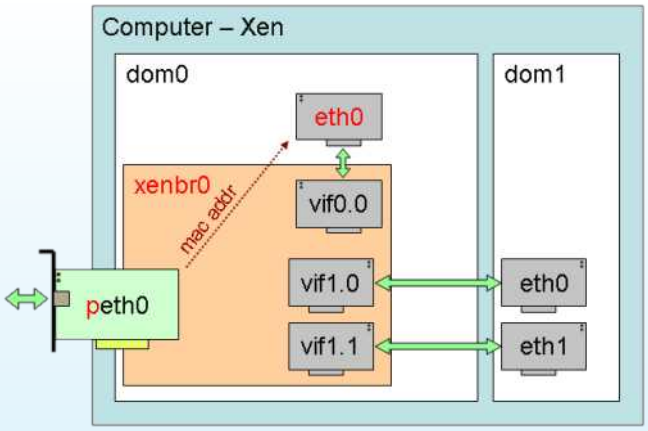
Első lépésben egy kis hálózati topológia. A tervezésben lényeges, hogy külön kapcsolat kezelje a szolgáltatást, és a szinkront/ellenőrzést. Erre a célra két olyan gépet képzeljünk el, aminek 2-2 hálózati interfésze van. A példa kedvéért lesz egy publikus lábunk `service0`, egy kapcsolatot ellenőrző láb `probe0`, és egy aggregált interfész a szinkron miatt `bond0`, mely a maradék két interfész összefogásából jön létre.

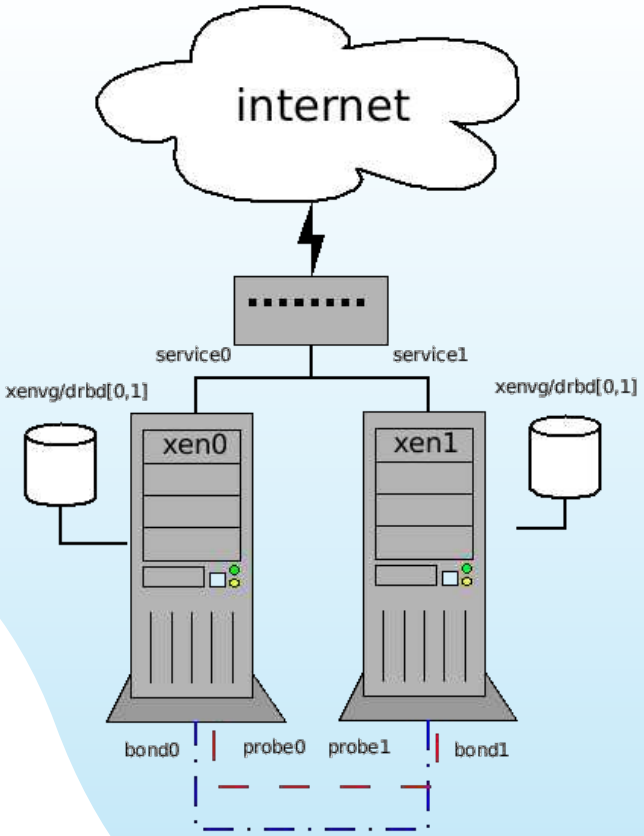
A diszkek esetében a eltekintünk a rendszerdiszk redundanciájától, ez mindenkinek saját felelőssége. A szinkronizálni kívánt partíciót egy `xenvg` PVG-ba tesszük, és a xen-tools LVM-ként hozza létre a diszkeket az alábbi logika alapján:

```
<hostname>-diszk
<hostname>-swap
```

Lássunk is hozzá az interfészek módosításához:

```
root@xen0#apt-get install ifenslave
```





Módosítsuk az */etc/networking/interfaces* fájlt:

```
auto bond0
iface bond0 inet static
        address 192.168.100.10
        netmask 255.255.255.252
        broadcast 192.168.100.11
        pre-up modprobe bonding
        up ifenslave bond0 eth2 eth3
        down ifenslave -d bond0 eth2 eth3
        post-down rmmod bonding

…
# Bonding parameters
options bonding mode=0 miimon=100
…
```

Valamint az alábbi kiegészítést fel kell venni az */etc/modprobe.d/options* -be:

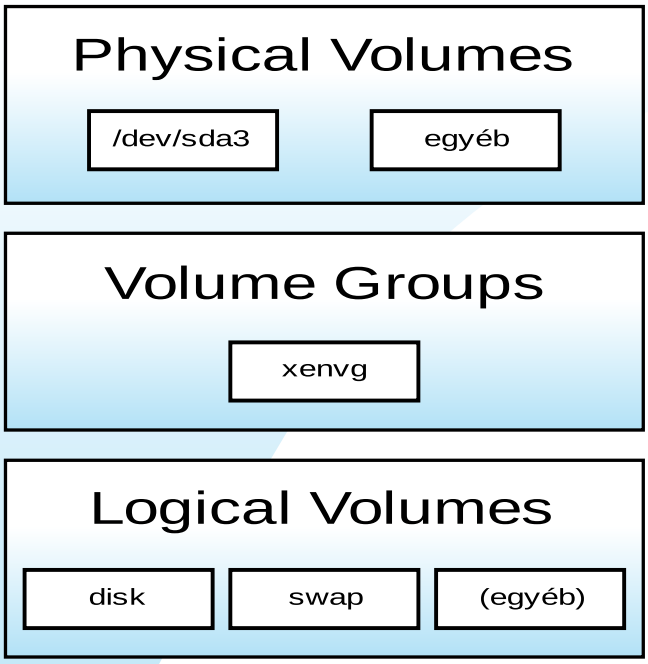
```
# PCI device 0x14e4:0x1659 (tg3)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
 ATTR{address}=="00:15:f2:e0:c0:73", ATTR{type}=="1", KERNEL=="eth*",
 NAME="service0"
# PCI device 0x14e4:0x1659 (tg3)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
 ATTR{address}=="00:15:f2:e0:c0:74", ATTR{type}=="1", KERNEL=="eth*",
 NAME="probe0"
```

Nevezzük át az interfészeket, a **/etc/udev/rules.d/70-persistent-net.rules** fájlban:

```
# PCI device 0x8086:0x1079 (e1000)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
 ATTR{address}=="00:04:23:ce:75:e6", ATTR{type}=="1", KERNEL=="eth*",
 NAME="service0"
# PCI device 0x8086:0x1079 (e1000)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
 ATTR{address}=="00:04:23:ce:75:e7", ATTR{type}=="1", KERNEL=="eth*",
 NAME="probe0"
```

Telepítsük a XEN csomagokat:

```
root@xen0#  apt-get  install  xen-hypervisor-3.2-1-amd64  xen-linux-system-2.6.26-1-xen-amd64  xen-
```




```
utils-3.2-1 xen-tools bridge-utils
```

Konfiguráljuk be a bridget:

```
root@xen0# brctl addbr xenbr0
root@xen0# brctl addif service0
```

Konfiguráljuk be a XEND -t a */etc/xen/xend-config.sxp* fájlban:

```
(xend-relocation-port 8002)
…
(xend-relocation-address '192.168.100.10')
…
(xend-relocation-hosts-allow
'192.168.100.11')
…
(network-script                'network-bridge
bridge=xenbr0')
```

Érdemes újraindítani a gépet, hogy felolvassa az új interfészneveket, az új xen patchelt kernellel bootoljon fel, és jobb ha odafigyelünk a bridge létrejöttére is, mert sajnos a gyakorlat azt mutatja, hogy sokszor ügyeskedni kell vele, mivel a xen3.0 és xen3.1/2 között sok változás volt, és ez gyakran okoz kavarodást. Ha úgy érezzük, hogy készen vagyunk, akkor ellenőrizzük le:

```
root@xen0# brctl show
bridge name    bridge id            STP enabled    interfaces
xenbr0         8000.0014c2547612    no      service0
```

Telepítsük az LVM -et:

```
root@xen0# apt-get intsall lvm2
```

Tételezzük fel hogy van egy szabad eszközünk, vagy partíciónk, amit teszem azt **sda3**-nak hívnak:

```
root@xen0# pvcreate /dev/sda3
…
root@xen0# vgcreate xenvg /dev/sda3
root@xen0# vgdisplay
  --- Volume group ---
  VG Name               xenvg
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  44
  VG Access             read/write
  VG Status             resizable
…
  VG Size               54.73 GB
  PE Size               4.00 MB
  Total PE              14011
  Alloc PE / Size       6176 / 24.12 GB
  Free  PE / Size       7835 / 30.61 GB
```

Érdemes módosítani az */etc/xen-tools/xen-tools.cfg* fájlt, hogy ne kelljen mindet CLI-ben megadni.

…

```
lvm = xenvg
…
gateway = 10.1.1.254
netmask = 255.255.255.0
broadcast = 0.0.0.255
…
passwd = 1
accounts = 1
…
mirror = http://ftp.hu.debian.org/debian/
serial_device = xvc0
```

Hozzuk létre a xen-instance-ot, nevezzük el xen-instance1-nek, mert ez fantáziadús név.

```
root@xen0# xen-create-image -hostname=xen-instance1 -ip=10.1.1.50
```

Nyomon követhető a telepítés minden kínja, a */var/log/xen-tools/xen-instance1.log* -ban. Ha elkészült akkor az */etc/xen/xen-instance1.cfg* fájlban lesznek megtalálhatóak a virtuális gép paraméterei. Valamint létrejön a */dev/xenvg/xen-instance1-disk* és a */dev/xenvg/xen-instance1-swap* LVM eszközök is. A továbbiakban ez lesz a lényeges számunkra. A DRBD segítségével mindkét device-t szinkronba tudjuk majd tartani. Aki nem ismerné, annak elmondom, hogy a DRBD lényegében egy szoftraid eszköz, ami TCP/IP protokollon képes a RAID1-et fenntartani.
Természetesen a másik gépen (xen1) is létre kell hozni a VG-t és mind az *xen-instance1-disk, xen-instance1-swap* LVM devicet is, úgy hogy az pontosan megegyezzen a forrás gépen lévő méretével, és a fenti lépéseket is el kell végezni értelemszerűen.

```
root@xen1# lvcreate -L 5G -n xen-instance1-disk
```

Ha ezzel készen vagyunk jöhet a DRBD installálás:

```
root@xen0# apt-get install drbd8-utils
```

A beállítások a következőek az */etc/drbd.conf* -ban, csak a lényeges részletek vannak kiemelve:

```
…
resource disk {
protocol C;
startup {
wfc-timeout 120; ## 2 min
degr-wfc-timeout 120; ## 2 minutes.
}
…
on xen0 {
address 192.168.100.10:7789;
device /dev/drbd1;
disk /dev/xenvg/xen-instance1-disk;
meta-disk /dev/xenvg/meta[0];
}
on xen1 {
address 192.168.100.11:7789;
device /dev/drbd1;
disk /dev/xenvg/xen-instance1-disk;
meta-disk /dev/xenvg/meta[0];
}
…
```

```
…
resource swap {
protocol C;
startup {
wfc-timeout 120; ## 2 min
degr-wfc-timeout 120; ## 2 minutes.
}
…
on xen0 {
address 192.168.100.10:7790;
device /dev/drbd2;
disk /dev/xenvg/xen-instance1-swap;
meta-disk /dev/xenvg/meta[1];
}
on xen1 {
address 192.168.100.11:7790;
device /dev/drbd2;
disk /dev/xenvg/xen-instance1-swap;
meta-disk /dev/xenvg/meta[1];
}
…
```

Hozzuk is létre a fent szereplő metaadatokat tároló eszközt, valamint inicializáljuk a DRB eszközöket:

```
root@xen0# lvcreate -L 1G -n meta xenvg
root@xen0# drbdadm create-md xen-instance1-disk
root@xen0# drbdadm create-md xen-instance1-swap
```

Nagyon fontos, hogy a DRBD eszközök közül csak az egyik gépen (a primer node-on, jelen esetben a xen0-n) legyen *Primary* a drbd státusz, mert a DRBD8 már képes *Primary/Primary* működésre, ami a mi esetünkben inkonzisztens diszk állapotot eredményez.

```
root@xen0# drbdadm /dev/drbd1 primary -o
root@xen0# drbdadm /dev/drbd2 primary -o
root@xen0# /etc/init.d/drbd status
drbd driver loaded OK; device status:
version: 8.0.13 (api:86/proto:86)
m:res  cs         st                 ds                 p  mounted  fstype
1:disk  Connected  Primary/Secondary  UpToDate/UpToDate  C         ext3
2:swap  Connected  Primary/Secondary  UpToDate/UpToDate  C         ext3
```

Annak érdekében, hogy a virtuális gépünk tisztában legyen a kialakult helyzettel, módosítanunk kell a xen-instance1.cfg leíró fájlt. Ez azért lényeges, mert a *xend* odafigyel arra, hogy csak akkor indítson el xVM-t ha a drbd diszk *Primary* állapotban van, sőt leállás után *Secondary* állapotba helyezi vissza. Átmásolva a módosított konfigurációt mindkét gépen el tudjuk indítani a virtuális gépet, de soha ne indítsuk egy mindkét hoszton egyszerre, mert ebben az esetben végérvényesen „tönkretehetjük” a diszket.

```
…
disk = [
    'drbd:xen-instance1-swap,xvda1,w',
    'drbd:xen-instance1-disk,xvda2,w',
]
…
```

(xen-instance1.cfg)

Ha mindez készen van kiadhatjuk a varázslatos és csodálatos **`xm migrate xen-instance1 192.168.100.2 -live`** parancsot. Ha minden jól megy akkor a drdb státuszt módosítja, az xVM-et a megjelölt gépre mozgatja, anélkül, hogy bárki bármit észrevenne. Ahogy a kedvenc filmemben mondják *„...és a film pörög tovább, senki nem vesz észre semmit...”*.

Ha mindez sikeresen megtörtént, akkor most itt az ideje, hogy automatizáljuk a migrációt. Ehhez HEARTBEAT-et fogunk használni. Az alapvető fájlok a *ha.cf*, *authkeys*, és a *haresources* (példákat találunk az */usr/sharre/doc/heartbeat* könyvtár alatt). Oda kell figyelni itt is, hogy ki kell jelölni egy elsődleges szervert a XEN instance-nak. Itt autoboot-ol a gép. Mint már említettem a legfontosabb, hogy ne fusson egyszerre a xen0-án és xen1-en, mert akkor inkonzisztens diszk állapotot érhetünk el. Ez ugyebár nem túl előnyös.

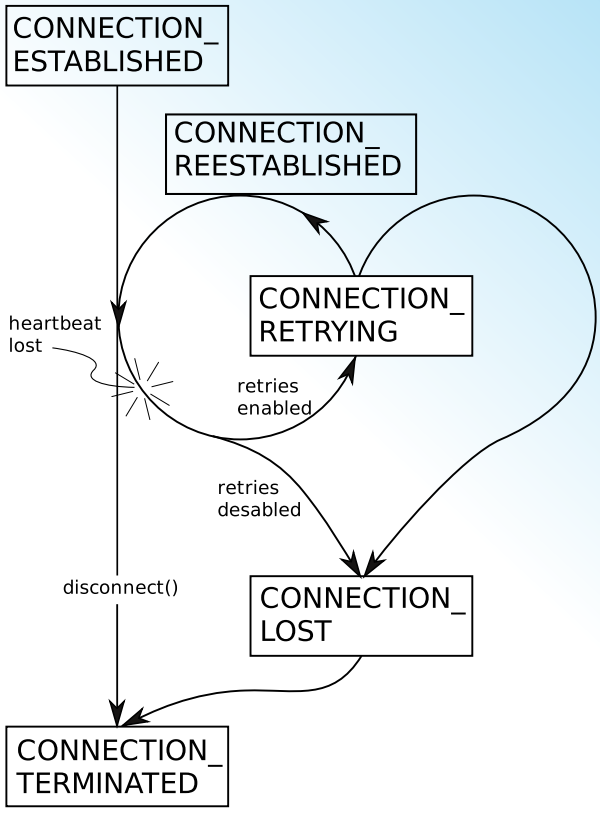
```
root@xen0# apt-get install heartbeat
root@xen0# cat /etc/ha.d/authkeys
auth 1
1 sha1 Fl0ssz1NE:)
```

Ezzel a kulccsal azonosítják egymást a szerverek. Győződjünk meg róla, hogy ugyanaz a jelszó van mindkét gépen. A ha.cf fájlban sok egyéb mellett meg tudjuk adni, hogy melyik interfészen kommunikáljon, ezt érdemes egy külön kábelen átfuttatni, így kizárható a hálózati hiba, ami egy részről jó, hiszen akkor is működőképes a gép, ha a szolgáltatás nem. Érdemes tovább finomítani a konfigot úgy, hogy vegyen figyelembe, mondjuk egy külső oldal, vagy szerver elérhetőségét is. Ez egy érdekes probléma.

A mi egyszerűsített *ha.cf* konfigurációnk mindkét *node*-on:

```
logfacility local0
```

```
udpport 694
keepalive 1
deadtime 10
warntime 3
initdead 20
ucast probe0 172.0.100.1
ucast probe1 172.0.100.2
auto_failback on
watchdog /dev/watchdog
debugfile /var/log/ha-debug
node xen0
node xen1
```



Definiálni kell a haresources fájlban a megadott node elhalása esetén érvénybe lépő parancsot.

```
xen0    xendomains-0
xen1    xendomains-1
```

Ezt a parancsot az /etc/ha.d/resources.d/ könyvtárban fogja keresni, így oda létre kell hozzuk azt. Másoljuk le a xendomains scriptet két példányban, majd ezeket módosítjuk, hogy a másik gépre migrálja a virtuális gépünket. Ezt mindkét node-on meg kell csinálni.

```
root@xen0# cp /etc/init.d/xendomains /etc/ha.d/resources.d/xendomains-0
root@xen0# cp /etc/init.d/xendomains /etc/ha.d/resources.d/xendomains-1
```

Módosítjuk a lock fájlt, és a script konfig fájlt az */etc/ha.d/resources.d/xendomains-1* és *2* ben:

```
…
LOCKFILE=/var/lock/xendomains[0|1]
XENDOM_CONFIG=/etc/default/xendomains[0|1]
…
```

Másolatot készítünk a szkriptről *xendomains-0* és *xendomains-1* névre, majd azt is átírjuk pár helyen:

```
…
XENDOMAINS_MIGRATE="192.168.100.[1|2] --live"
XENDOM_CONFIG=/etc/default/xendomains[0|1]
XENDOMAINS_SAVE=
XENDOMAINS_SHUTDOWN_ALL=
XENDOMAINS_RESTORE=false
XENDOMAINS_AUTO=/etc/xen/auto/xen[0|1]
XENDOMAINS_AUTO_ONLY=true
…
```

Már csak helyére „dombjuk” az xVM konfigurációit, az */etc/xen/auto/xen**[0|1]*** könyvtárakba és elindítjuk a HEARTBEAT-et, és reméljük a legjobbakat.

```
root@xen0# mkdir /etc/xen/auto/xen0 && mkdir /etc/xen/auto/xen1
root@xen0# ln -s /etc/xen/xen-instance1.cfg /etc/xen/auto/xen1/
root@xen0# update-rc.d -f xendomains remove
root@xen0# xm shutdown xen-instance1
root@xen0# /etc/init.d/heartbeat start
```

Befejezés

A fent leírtakat nem tutorialnak szántam, csupán segédletnek. Igyekeztem a leginkább a valós életben is működő értékeket használni, de lehet hogy néhány helyen korrigálni, vagy egyéb beállításokat is használni kell, hogy a leírtak alapján működjön a migráció. Ha valaki konkrét tutorialt szeretne a témában annak javaslom látogasson el a http://www.asplunk.nu weboldalra. Az ott leírtak ihlettek arra, hogy magam is kipróbáljam a *live-migration* ezen formáját, és nagyban segített a leírás elkészítésében is. További kellemes napot mindenkinek és sok sikert azoknak, akik nekilátnak kialakítani a fent leírt rendszert!

Csíkos Bálint

http://www.asplund.nu/xencluster/xen-cluster-howto.html
http://www.cl.cam.ac.uk/research/srg/netos/papers/2005-migration-nsdi-pre.pdf
http://etbe.coker.com.au/2007/08/13/ethernet-bonding-on-debian-etch/
http://en.wikipedia.org
http://www.ghostintheshell.tv/
http://www.gatzet.info/
http://redhat.com/
http://docs.sun.com

Tudtad-e?

RETRÓ

Európa egyik legnagyobb és kétségtelenül legkülönlegesebb számítástechnika-történeti múzeumát hozzák létre a Szegedi Informatikai Gyűjteményből[1]. 1992-ben többek között Kovács Győző és Muszka Dániel kezdeményezésére jött létre a Magyar Informatikatörténeti Múzeum Alapítvány[2], amely célul tűzte ki a divatjamúlt számítógépek megőrzését[3]. Ezek a rendszerváltásig hivatalosan csak a KGST-országokból kerülhettek be Magyarországra, vagy itthoni fejlesztésű gépek lehettek, de az embargó ellenére néhány nyugati eredetű szerkezet is érkezett a legvidámabb barakkba. Közülük 1972-től máig több mint 200 tonnányi számítógép, illetve periféria gyűlt össze. Az 1995-ben a Szegedi Egyetem által befogadott gyűjtemény jelenleg mintegy 2000 négyzetméteren tekinthető meg előzetes bejelentés után a szegedi volt laktanya épületében.

A Neumann János Számítógéptudományi Társaság folyamatos támogatása mellett működő Alapítvány 2007 óta már pályázatokon is részt vehet. Az Országos Műszaki Múzeummal, a Szegedi Egyetemmel, és Szeged Megyei Jogú Város Önkormányzatával közösen azt tervezik, hogy a gyűjteményt 2010-ben végleges helyére, az Agóra Szeged Pólus centrumban megnyíló Csodák Palotájába költöztetik[4], ahol az ősgépek közül minél többet megpróbálnak majd ismét működésre bírni.

[1] http://www2.u-szeged.hu/infmuz/Gyujtemeny.htm
[2] http://www.machines.hu/cgi-bin/machines/new/cikk.cgi?cikk=266
[3] http://retropages.uw.hu/Download/Szeged_muzeum.pdf
[4] http://www.portalbin.njszt.hu/2009/Bohus_Mihaly_eloadasa_01.ppt

spamszűrés

Clapf

Hulladékfeldolgozás nyílt forrású programmal - 2. rész

Az előző számban egy rövid áttekintés után telepítettük a Clapf spamszűrőt és kipróbáltuk csak vírusirtó interfészként a SpamAssassinnal együttműködve. Láttuk, hogy szükség volt egy master.cf trükkre is, azonban eltelt 2 hónap és most már a Clapf az --enable-spamc-emul configure opció hatására maga beszélget a spamdvel és nincs szükség a spamcre. Ebben az írásban viszont már bevetünk mindent és megnézzük, mire képes egy statisztikai spamszűrő. A példákban gyakran fogok hivatkozni egy Béla nevű felhasználóra (username: bela, uid: 1001), aki mindig a kialakított rendszer egyik felhasználója, akinek az e-mail címét a Clapf védi.

Az adatbázis

Szó volt arról, hogy a *Clapf* adatbázisban tartja a tokeneket. Az adatbázis bármi lehet, amiből a tokenekhez tartozó számlálókat le lehet kérdezni, de mivel egy átlagos levél esetén több száz tokent is meg kell nézni, ezért a megfelelően gyors működés érdekében a statisztikai spamszűrők jellemzően valamilyen bináris adatstruktúrát használnak, pl. Berkeley DB, Tokyo Cabinet, vagy pedig hagyományos relációs adatbázist, mint pl. SQLite3 vagy MySQL. A clapf egyébként ez utóbbi kettőt támogatja hivatalosan. A fordításkor tehát meg kell adni a `'--with-tokendb='` configure opció után vagy a `mysql` vagy az `sqlite3` értéket.

Amikor a clapf egy levelet kap, akkor meghatározza, hogy melyik felhasználónak szánták, illetve hogy mely token halmazt használja a valószínűségértékek kiszámításához. Ha a *clapf.conf*-ban a `group_type=0` szerepel, akkor minden felhasználó egy közös, ún. globális adatbázison osztozik. Ebben az esetben nincs lehetőség[1] arra, hogy ugyanazt a levelet az egyik felhasználó spamnek, míg a másik jó levélnek tekintse.

A clapf azonban lehetővé teszi, hogy az egyes felhasználók testre szabják a token adatbázisukat. Ez a `group_type=1` beállítással érhető el, amelynek hatására a clapf a globális és a felhasználó saját token adatbázisának unióját képezi, és azzal dolgozik. Ez a gyakorlatban annyit jelent, hogy pl. a `'pharmacy+online'` token kétszer is szerepelni fog a *t_token* táblában. Egyszer 0-s uid-del, azután 1001-es uid-del a globális adatbázisban, illetve a felhasználó tokenadatbázisban, amint az az alábbi példában is látható. A `where` kifejezésben szereplő szám a szöveges token numerikus kódolt értéke. Ezzel a megoldással 8 byte-on is elférnek a tokenek, egy rekord mérete pedig 20 byte körül van.

```
mysql> select * from t_token where token=13961784965195248826;
+---------- -+------+-------+-------+------------+
| token  | uid | nham | nspam | timestamp |
+----------------------+------+-------+-------+------------+
| 13961784965195248826 | 0 | 2 | 1 | 1241698716 |
| 13961784965195248826 | 1001 | 0 | 7 | 1241690266 |
+----------------------+------+-------+-------+------------+
2 rows in set (0.00 sec)
```

Ha Béla csak a globális adatbázist használja, akkor – ebben a példában – nem ismeri fel spamre jellemző tokenként a `pharmacy+online` kifejezést, mivel kettő jó levélben szerepelt, és csak egy spamben. Béla azonban tanította a spamszűrőt, így a saját token halmazában 7-szer szerepelt spamként ez a kifejezés, így már (2 ham vs 8 spam) spamre jellemző mintaként azonosítja a program.

Közös token adatbázist akkor érdemes használni, ha a felhasználók hasonló mintájú levelezést folytatnak, pl. egy cég esetén, vagy ha csak egy korlátozott méretű diszket akarunk a tokenek számára biztosítani, esetleg ha beérjük 99% körüli pontossággal. Ha azonban egészen eltérő az egyes felhasználók levelezése, mondjuk egy szolgáltatónál, akkor érdemes

bevetni a tokenek testreszabásának lehetőségét. Ekkor azonban figyeljünk oda, hogy az SQL tábla eléggé meghízhat, de ez a purge* scriptekkel kezelhető szinten tartható.

Az adatbázist az első alkalommal létre kell hozni, és inicializálni kell a tartalmát. Ez a választott token adatbázistól függően az alábbi utasítások egyikével tehető meg:

```
mysql --defaults-file=/usr/local/share/clapf/.my.cnf < /usr/local/share/clapf/db.sql
```

vagy

```
sqlite3 /var/lib/clapf/data/clapf.sdb < /usr/local/share/clapf/db.sql
```

Ha a spameket már eddig is egy külön mappába gyűjtöttük össze, akkor az inicializálást összeköthetjük a kezdeti tanítással. Ehhez használjuk az előzőek helyett az alábbiak egyikét:

```
/usr/local/libexec/clapf/db_init_mysql.sh HAM-mbox SPAM-mbox /usr/local/share/clapf/.my.cnf
```

vagy

```
/usr/local/libexec/clapf/db_init_sqlite3.sh HAM-mbox SPAM-mbox /var/lib/clapf/data/clapf.sdb
```

A (H)SPAM-mbox argumentum az vagy egy mbox formátumú fájl, vagy pedig egy könyvtár, amelyben maildir formátumú e-mailek vannak. Fontos: bármely statisztikai spamszűrőre igaz, hogy mind ham, mind pedig spam levelekkel tanítani kell. A számuknak nem kell megegyezniük, de jó, ha hasonló nagyságrendben szerepelnek. Tapasztalataim szerint pár ezer levéllel tanítva bőven 99% feletti pontosságot érhetünk el a programmal.

A clapf nemcsak a tokeneket tartja nyilván, hanem a felhasználókat, és azok e-mail címeit is. Ezeket szintén adatbázisban tartja, SQL esetén ez egy másik táblát jelent az adatbázisban, de a felhasználó adatokat akár LDAP címtárban is tarthatjuk, így egy meglévő Active Directoryhoz is integrálható a clapf. A `'--with-userdb='` configure opcióval választhatunk a lehetőségek közül.

Az inicializálás során létrejön egy `'admin'` nevű felhasználó `'admin'` jelszóval. Ne felejtsünk el neki jelszót változtatni a webui-n – erről még bővebben szó lesz később.

Egy kisebb rendszer SQLite3-mal

Ennyi fejtágítás után nézzünk meg egy kisebb rendszert, ahol mind a tokenek, mind pedig a felhasználói adatok SQLite3 adatbázisban vannak. Fordítsuk le a programot az alábbi parancsokkal (részletes telepítési útmutató a projekt honlapján található):

```
./configure --enable-clamd \
            --with-clapf-user=clapf \
            --localstatedir=/var \
            --with-userdb=sqlite3 \
            --with-tokendb=sqlite3
make
su -c 'make install'
```

Az SQLite3 nagyon gyors adatbázis, azonban mivel mindenki ugyanazt az állományt használja, ezért nagyobb forgalom esetén előjöhet a konkurens hozzáférés problémája. A clapf ugyanis, miután kiszámolta a levél spam-valószínűségét, frissíti a levélben szereplő tokenek időbélyegét a *t_token* táblában. Ez azért hasznos, mert így nyomon tudjuk követni, hogy mely tokeneket nem használjuk már, azok pedig minden további nélkül törölhetőek.

Bizonyos környezetben azonban ezt el lehet hagyni, és ha a konfigurációs fájlban kikapcsoljuk a tokenek időbélyegének a frissítését (`update_tokens=0`), akkor konkurens hozzáférés esetén is nagyon gyors működést kapunk (néhány ms levelenként), mert ebben az esetben csak olvasásra kell megnyitni az adatbázist, és nem kell zárolással bajlódni. Ebben az esetben viszont ne futtassuk az éjjelente szokásos törlő scriptet (*purge-sqlite3.sh*), amely az előbb említett takarítást elvégzi.

Nagyobb rendszer MySQL-lel

Következő lépésben építsünk egy nagyobb környezetbe szánt kiszolgálót[2], amely MySQL adatbázist használ. Telepítés után ne felejtsük el beállítani az adatbázis elérésének paramétereit.

```
./configure --enable-clamd \
            --with-clapf-user=clapf \
            --localstatedir=/var \
            --with-userdb=mysql \
            --with-tokendb=mysql \
```

```
--with-store=nfs
```

Nagyobb környezetbe nem elég egyetlen antispam-kiszolgáló, tegyünk be rögtön kettőt. Nem csak Mátrix-filozófia van, hanem van clapf-filozófia is, aminek az a lényege, hogy a telepítés után a rendszergazda minél kevesebbet dolgozzon, és amit a felhasználók is meg tudnak tenni, azt hadd tegyék meg ők. Amikor Béla már a harmincadik spamet kapja aznap, nem csoda, ha frusztrált lesz, hogy semmit nem tud tenni – a DEL gombra könyöklésen kívül. Nem így a statisztikai szűrők esetén. A clapf ugyanis lehetőséget ad arra, hogy a felhasználók is tanítani tudják az adatbázist. Ehhez szükség van az eredeti levélre, amit a clapf - ha használjuk a **'--with-store'** opciót – eltárol egy *queue* könyvtárban. Amikor a felhasználó tanítja a szűrőt, akkor a clapf kihámozza belőle az eredeti levél azonosítóját, és így megtalálja az eredeti, módosítatlan példányt.

Egy probléma azért akad: ha kettő vagy több gép dolgozza fel a leveleket, és egy fel nem ismert spam az 'A' gépen kerül tárolásra, de a tanítás a 'B' gépre érkezik, akkor gond van. Itt jön a képbe az NFS. A két spamszűrőt futtató gép mellé szükséges egy harmadik, amelyet az 'A' és 'B' gép felmountol a queue könyvtár (azaz pl. a */var/lib/clapf/queue*) alá, és oda helyezi el az eltárolt levelet. Ha csak egy gépen akarunk clapf spamszűrőt futtatni, akkor a **'--with-store=fs'** a megfelelő választás, mert ekkor a helyi gép ugyanazon partícióján tárolja a leveleket, és a **link()** függvény nyilván gyorsabb, mintha hálózaton kellene másolni. A **'--with-store=nfs'** opciót kell abban az esetben is használni, ha a (*/var/lib/clapf/*)*tmp* könyvtár a helyi gépen van ugyan, de másik partíción, mint a queue könyvtár, mert a **link()** rendszerhívás nem működik partíciók között. Érdemes TCP protokollt használni a spamszűrőt futtató gépeken, és rw, async, no_root_squash, no_subtree_check opciókkal kiajánlani az nfs klienseknek:

```
mount -o proto=tcp nfs.aaaa.fu:/export /var/lib/clapf/queue/
/etc/exports:
/export 1.2.3.4(rw,async,no_root_squash,no_subtree_check)
```

Tanulni jó!

A tanítás egyébként nem túl bonyolult a felhasználók számára. A tévesen kategorizált levelet el kell küldeni egy speciális e-mail címre. A fel nem ismert spameket a spam@domain címre, a fals pozitív hibákat (azaz a tévesen spamként osztályozott jó leveleket) a ham@domain címre. Még egy követelmény van: a feladó levelezőkliensében a clapf számára ismert e-mail cím legyen beállítva. Ha ez utóbbi valamiért nem járható, mert pl. a felhasználó a bela@domain címre kapja a leveleit, de a kliensében bela@masikdomain van beállítva, akkor a tanítandó leveleket a bela+spam@domain, illetve a bela+ham@domain címekre lehet küldeni. A tanító címekre érkező leveleket a clapf démon elkapja, feldolgozza, de nem küldi tovább.

Mivel azonban most több gépen futtatjuk a spamszűrőt, ezért ugyanazt a MySQL adatbázist több gépről kell elérnünk, ami lehet pl. az NFS szervert futtató gépen. Elvileg az is járható út, ha MySQL replikációt használunk, azonban ehhez az SQL 'írásműveleteket' (update, insert, delete) más gépen (a master szerveren) kellene elvégeznünk, mint az olvasást (select), ezt azonban a clapf nem támogatja a cikk írásakor. Egy másik lehetőség lehetne a MySQL cluster használata, azonban ez túlmutat a spamszűrő finomhangolás keretein, és a legtöbb környezetben valószínűleg túl nagy ágyú lenne a feladatra.

LDAP

A felhasználói információkat LDAP címtárban is tarthatjuk, ha a **'--with-userdb=ldap'** opcióval fordítjuk le a programot. Az LDAP szerveren szükséges a *qmail.schema* és a *clapf-policy.schema* állományok importja (*l. slapd.conf*). Az előbbi a *qmail*hez készített séma módosított, clapf-specifikus attribútumokkal kibővített változata, az utóbbi pedig a házirend-szabályok leírásait tartalmazza. Ha *Active Directory*nk van, akkor nekünk kell elkészíteni a *qmail.schema*-ban szereplő extra attribútumokat, ld. a projekt honlapját bővebb információkért. Végül adjuk meg a kapcsolódás paramétereit a clapf.conf állományban. Példa LDIF fájlokat szintén a clapf weboldalán találhatunk.

Feketelisták

A clapf feketelisták használatát is támogatja, amely lehet egy hagyományos lista, mint pl. a *Spamhaus*, de lehet URL feketelista is, mint pl. az *uribl* vagy a *SURBL*. Míg az RBL-listák megszokott használata egy meglehetősen fekete-fehér világképet erőltet ránk (mivel egy adott IP-címről érkező összes levelet vagy eldobatja vagy beengedteti velünk), addig a clapf intelligens módon alkalmazza az RBL listákból kinyerhető információt. Ha a levél spam valószínűsége nagyobb egy határértéknél (*max_ham_spamicity*), de kisebb, mint a spam határérték (*spam_overall_limit*), tehát a spamszűrő bizonytalan,

hogy a levél jó vagy sem, akkor lekérdezi a beállított feketelistákat, hogy mi a véleményük a velünk beszélgető SMTP kliensről, illetve a levélben szereplő web oldalakról? A DNS kérések eredménye egy-egy rossz token, amennyiben pozitív választ (általában 127.0.0.x) adnak. Így a majdnem spam levél valószínűsége jó eséllyel felemelkedik a spam határ fölé, és a clapf megfelelően értékelheti a levelet.

Tegyük fel, hogy az egyik levelezőpartnerünk szolgáltatójának SMTP szerverei feketelistára kerültek, mert az egyik kereskedőjük úgy gondolta, hogy a karácsonyi értékesítést megfejeli egy rózsaszín szerződéssel. A clapf, ha a statisztikai modul szerint a levél jó, akkor nem bántja azt, így a rendszeres levelezőpartnereink leveleit nem kell féltenünk a programtól. Ha azonban egy rövid *smswarez* spamet kapunk, amiben Brigi puszil minket, és még éppen a spam határérték alatt van a valószínűsége, akkor a feketelisták lesújtó véleménye a levelet a határérték fölé emeli, így segítve a spamszűrönkön.

Bár a feketelisták kiegészítő használata javíthatja a pontosságot, ennek azonban ára van: a DNS kérések (főleg ha sok URL van a levélben) válaszideje megnöveli a levél feldolgozásának idejét, azaz tovább tart a spamszűrőnek, amíg eldönti, hogy mi legyen a levél sorsa. Megfelelő tanítás esetén azonban nincs szükségünk sokszor erre a mankóra.

Házirendek

A 0.4.0-tól kezdve a clapf támogatja a házirendek használatát. A konfigurációs fájlban egy nagy halom paraméter van, amelyet akár felhasználónként is állíthatunk. Alapesetben mindenki a nullás, azaz alapértelmezett csoportban van, ami a clapf.conf beállításait jelenti.

Tegyük fel, hogy Béla azt szeretné, ha a spam leveleket *[SPAM]* előtaggal jelöljük meg neki. Ebben az esetben hozzunk létre egy új házirend csoportot (*policy group*), amely csak a *spam_subject_prefix* paraméterben tér el az alapértelmezettől, és vegyük fel a kívánt felhasználókat az új csoportba. A felhasználók és házirendek kezelése pár kattintással megoldható a később ismertetett *webui*-ban, ami a clapf webes felületű menedzsment rendszere, de akár egy alkalmas SQL utasítást vagy LDIF állományt is használhatnak a haladó adminisztrátorok. A különféle házirendekkel azt is megtehetjük, hogy az egyik felhasználónak csak megjelöljük a spam leveleit, míg a másiknak karanténba zárjuk. Beállíthatjuk a használni kívánt feketelistákat, a spam valószínűség határértéket, de még azt is, hogy egyáltalán használni akarjuk-e a clapf spamszűrő modulját az adott csoportban. A lehetőségeknek csak a fantázia szab határt.

E-mail címlisták

A clapf lehetőséget ad, hogy bizonyos e-mail címeket fekete- vagy fehérlistára vegyünk fel. Ha egy címet (vagy mintát) a feketelistára teszünk, akkor az attól a feladótól származó leveleket a clapf naplózás után eldobja, míg ha fehérlistára tesszük, akkor minden körülmények között (kivéve ha vírust küldött) elfogadja. Gondoltam a tréfás kedvű felhasználókra is, akik ugyanazt a címet felveszik mindkét listára: a clapf a fehérlistát részesíti előnyben.

Ha pl. Bélát és az összes Gmail címet fel akarjuk venni, akkor ezt az alábbi módon tehetjük meg. A '$' szimbólum azt akadályozza meg, hogy a @gmail.com.*spammerdomain.com* domainből érkező spamek átcsússzanak a szűrőn.

```
bela@aaaa.fu
@gmail.com$
```

A következő rész tartalmából

A befejezésben megnézzük a már említett webui-t, egy-két tippet, trükköt, illetve finomhangoljuk a spamszűrőt a még nagyobb teljesítmény érdekében, és eloszlatunk pár félreértést a statisztikai szűrőkkel kapcsolatban.

Sütő János

1. Azonban ebben az esetben is lehetőség van arra, hogy a felhasználó az e-mail cím feketelistájára felvegye a feladót, és így megszabaduljon egy bizonyos levéltől.
2. Természetesen akár egyetlen felhasználót tartalmazó rendszeren is használhatunk MySQL adatbázist a spamszűrővel.

telenet

VoIP Linux alatt

Most, hogy a gazdasági válság miatt minden forintot érdemes megfogni úgy magánszemélyként, mint cégként, előtérbe kerülnek a költségcsökkentő megoldások. Arányaiban legjobban talán a telekommunikációval tudunk spórolni, hiszen egy helyesen megválasztott internetes telefonszolgáltatóval nemcsak a havidíjat spórolhatjuk meg, de akár 4 forint 90 fillérért is – másodperces elszámolás mellett – elérhetjük a magyar vezetékes irányokat vagy a legtöbb EU-s ország vezetékes vonalát.

Mi kell ahhoz, hogy igénybe vehessük a *VoIP* lehetőségeit? Én általában ATA adaptert vagy asztali *VoIP telefont* szoktam ajánlani, de ez ugye 15 és 30 ezer forint közötti összeg, ami nem kevés. Ezt a kezdeti kiadást megtakaríthatjuk, ha softphone-t használunk, vagyis számítógépen futtatható telefonprogramot. Hátránya csupán annyi, hogy a számítógépnek bekapcsolva kell lennie. Persze ez irodai használatnál nem gond, pláne ha a *VoIP* szolgáltató nyújt hangposta szolgáltatást is.

Protokollok

Természetesen többféle protokollt is használhatunk, azonban ez behatárolja a szolgáltatót is. Például a *Skype*-ot nem kell bemutatnom. Saját zárt protokollal dolgozik és saját klienssel rendelkezik, így erre kár is több szót vesztegetni. Sokkal érdekesebbek számunkra a nyílt protokollok, úgy mint a *SIP* (Session Initiation Protocol) és az *IAX* (Inter-Asterisk eXchange). Az előbbit inkább VoIP végpontoknál, az utóbbit *asterisk*es központok összekapcsolásához használják. Nem szabad azonban az *IAX*-ről sem megfeledkeznünk, mert néhány kliensprogram ezt is támogatja a *SIP* mellett.

Kodekek

A másik érdekes kérdés a használt kodek. Ugyanúgy, ahogy egy filmnél vagy zenénél, itt sem mindegy, melyiket használjuk. Egyfelől függ a szolgáltatótól (attól, hogy mit kínál), másfelől függ a lehetőségeinktől (jellemzően a feltöltési sávszélességünk szabta korlátoktól). A szolgáltatók mindegyike támogatja a G.711u és G.711a formátumokat, melyek irányonként 64 kbit hasznos adatot visznek át másodpercenként. Számos szolgáltató támogatja még a GSM kodeket is, mely 13 kbit/s sávszélességű. Emellett persze vannak megoldások, amelyeknél figyelembe kell venni a licenceket és rendszerint fizetősek, ilyen például a G.729 (8 kbit/s) és a G.723.1 (5,3 vagy 6,3 kbit/s). (Ezen licencköteles kodekeket a szoftveres telefonok érthető okok miatt nem szokták támogatni.) Természetesen nemcsak ezek a kodekek léteznek, de ezek a leggyakrabban használtak.

Fontos megjegyezni, hogy forgalmunkhoz minden esetben hozzá kell még számolni a csomagfejlécet, így például a 64 kbit/sec G.711u-hoz minimum 80 kbit/sec ajánlott, de inkább egy picivel több.

A programokat Ubuntu 9.04-en próbáltam ki, a saját Ephone-os azonosítómmal, ezért kockáztam ki néhány képernyőkép idevágó részét.

Portforward

Gyakori *VoIP*-os hiba, hogy a felhasználó nem állítja be a routerén a porttovábbítást (portfoward, virtual server, stb.) Ez kimenő hívásokhoz nem mindig kell, bejövőek fogadásánál azonban már nagyon is. Ami minden esetben szükséges, az az, hogy a SIP portját (jellemzően 5060/5061 TCP vagy UDP port, de több softphone esetén mást is állíthatunk) továbbítsuk a megfelelő belső hálózati IP cím felé. Így a szolgáltató szervere tudja jelezni a programunk felé a bejövő hívást. Ha esetleg

valamelyik irányba nincs hang – akár kimenő, akár bejövő beszélgetéseknél –, akkor még az *RTP* portok forwardolása is szükséges (jellemzően 16000–22000 tartomány). Ez azonban nem mindig van így: a Linksys WRT54G routeremnek például tökéletesen elég a SIP port forwardja is. (Ez nem hinném, hogy a router gyártójától függ, vagy legalábbis nem közvetlenül.)

Kliensek

Ekiga

Az *Ekiga*-t Ubuntu csomagból telepítettem fel. A SIP-es fiók hozzáadása egyszerűen zajlik, csupán meg kell adnunk a SIP-es adatainkat a Szerkesztés -> Felhasználói fiókok menüben. A név mezőbe szinte mindegy, hogy mi van, a Regisztrátor a szolgáltató SIP szervere, a Felhasználó és a Felhasználó hitelesítése jellemezően a telefonszám.

Az *Ekiga* elvileg képes videóhívásra is, amennyiben a kernelünk támogatja a webkameránkat, azonban szolgáltató hiányában ezt nem tudtam kipróbálni.

Ha esetleg a kedves Olvasónak ismerős lenne az *Ekiga* felülete: korábban *Gnomeeting* néven futott.

Linphone

Következőnek a *Linphone*-t szerettem volna bemutatni, de sajnos az Ubuntu-s verzióval komoly bajok voltak. Terminálablakból indítva csúnya hibákat dobált. Minthogy próbálom egyszerű felhasználóként megejteni a teszteket, nem hiszem, hogy feladatom lenne a hiba okát kinyomozni.(Nekem Ubuntu Jaunty-n a 2.2.1-es verzió elundult zokszó nélkül, de ennél tovább nem próbáltam.)

Kphone

A következő a *kphone*. Kicsit puritán, de ez is kezelné a webkamerámat, ha lenne. A felhasználói beállításokat itt a menü alatti gombra kattintva érhetjük el. A *SIP* URL felhasználói része és a Hitelesítési Felhasználónév jellemezően a telefonszámunk, míg a *SIP* URL host része és a Kimenő Proxy a szolgáltató címe. Az Ekigánál lényegesen puritánabb, így kissé fenntartásokkal ajánlanám csak.

Twinkle

Majdnem a teszt végén, de be kell mutatnom egy másik (a kphone nem szintén Qt-s? – Oszkár) Qt-s játékost is. A *Twinkle* nem egy pehelysúlyú versenyző. Azontúl, hogy a Twinkle néz ki talán a legjobban, talán a leginkább testre szabható kliens. A „gyári” beállításokkal is remekül megy, de bármi egyedi állítás már erősen pilótavizsgás. A Twinkle-nek van azonban még egy érdekes funkciója: képes a hanganyagot titkosítani. Természetesen ennek csak akkor van értelme, ha a túloldal meg tudja fejteni.

Kiax

Kissé kakukktojás a *Kiax*, mert Magyarországon szinte nincs olyan szolgáltató, aki nyújtana *IAX*-ot, azonban otthoni asteriskünk próbálgatásakor jól jöhet. Semmi felesleges sallang, de mégis kellemes a látványa a szemnek . Sajnos csak az *IAX*-et támogatja, a *SIP*-et nem. Azért is kakukktojás, mert nem az Ubuntu-val telepítettem fel, hanem a Sourceforge-ról töltöttem le.

Hazudnék, ha azt mondanám, hogy csak ennyi softphone van Linux alá. A zárt forráskódúakat például garantáltan kihagytam, mert azok jó eséllyel csak i386 architektúrára érhetőek el (ezt erősen kétlem, az amd64 azért csak támogatott platform), márpedig ha megnézzük, hogy mondjuk egy Debian hányféle architektúrát támogat, akkor világosan látszik, hogy ha létszámra nem is, de darabszámra az i386 a jéghegy csúcsa (na igen, de ki használ deszktopként nem i386/amd64 architektúrát, másfelől, hogy lefordult a Debianos fiúknak még sajnos ne jelenti, hogy működik is).

Általánosságban elmondható, hogy szinte az összes program remekül vizsgázott, igaz, csak G.711u kodekkel használtam. Egyetlen program mellé nem tudnám letenni a voksomat, de ha mindenképp muszáj lenne, akkor leginkább az *Ekiga* és a *Twinkle* közül választanék.

Medve Zoltán

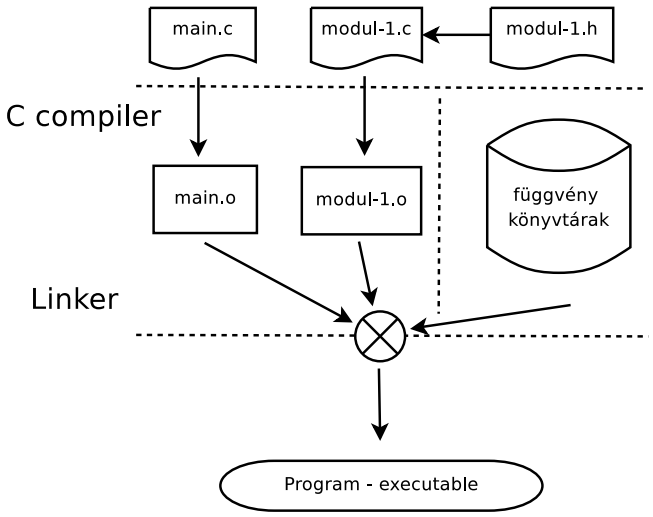
Hello World 5.

„MÚKODJ!” - azaz a *make utility* és a *Makefile*

A programozási munka során mindig eljön az a lépés, amikor a forráskódot a fordítóprogrammal le kell fordítanunk, ahogy azt az eddigi leckékben is láttuk. Akik az elmúlt 10-15 évben foglalkoztak programozással, azok gyakran ebből a folyamatból annyit ismernek, hogy kis kék bigyót kell megnyomni először majd a kis pirosat és már fut is a lefordított program. Mi nem akarunk ilyen éteri magasságokban röpködni a programfejlesztéssel, ezért megvizsgáljuk közelebbről a fordítási folyamatot és azokat a segédeszközöket, melyekkel parancssorból mindezt roppant kényelmesen el tudjuk végezni. Ezek lesznek – többek közt – a make utility és a Makefile.

A fordítás folyamata

A programok már régóta nem egy darab monolitikus massza módjára készülnek, hanem logikus egységenként darabokra bontjuk, illetve a már mások által jól megírt ilyen darabokat használjuk fel. Amikor a saját program darabjainkat használjuk az általában csak annyit jelent, hogy nem egy darab óriási méretű forrásállományt írunk hanem több file-ba szegmentáljuk a kódot, így egy kisebb változtatás, javítás miatt nem szükséges az egész programot újra fordítani csak a szükséges részeit. Arról nem beszélve, hogy így többen is dolgozhatunk egy programon zökkenőmentesen. A mások által megírt program részek általában általánosabb jellegű kódok, ezeket függvénykönyvtárakba szervezik, ezek a library-k, ezeket nem kell újrafordítanunk mert binárisan állnak rendelkezésre, csak hozzácsatolnunk más szóval linkelnünk a saját programunkhoz.



A fordítási parancsok

Amint láttuk az eddigiek során is, a fordítás parancsa a fordítóprogram meghívásából és a szükséges paraméterek átadásából áll. A paraméterek lehetnek a fájl nevek illetve a fordítási opciók. Ha megnézünk néhány ilyen parancsot, felfedezni vélünk néhány „kaptafát”, azaz közös elemeket, módszereket ezen parancsokban. Felmerül a kérdés, hogy nem lehetne ezeket valami egyszerű, egységesített módon kezelni? A válasz természetesen igen, sőt olyannyira, hogy a GNU rendszer első elkészült programjainak egyike a make utility volt, hogy a fordítási folyamat – az akkoriban igen szűkös – erőforrásokat minél hatékonyabban hasznosítsa.

A make használata

A make utility-t roppant könnyen tudjuk használni, parancssorba begépeljük, hogy make és megnyomjuk az ENTER-t. Ha ezt rögtön ki is próbáljuk, akkor valószínű az alábbi üzenetet kapjuk a rendszertől:

```
labor@otthon:~$ make
make: *** No targets specified and no makefile found.  Stop.
```

Vajon miért? Azért, mert a make utility nem egy gondolatolvasó varázseszköz, mely kitalálja, hogy mi mit is szeretnénk, hanem pontosan meg kell neki mondanunk, hogy milyen működést várunk el tőle. Ezen elvárásainkat egy file-ban közöljük a programmal, melynek neve lehet GNUmakefile, makefile, illetve Makefile, a make ebben a sorrendben keresi azon alkönyvtárban, ahol kiadtuk a parancsot. Itt még tegyünk egy kis kitérőt és ejtsünk pár szót a make-ről. Ez a program valójában messze nem csak arról szól, hogy C programokat könnyebben le tudjunk fordítani, ez egy elég általános utility

melyet legfőképpen erre használunk. Elsődleges feladata a programnak az, hogy észlelje egyes állományok megváltozását és ezen megváltozások alapján shell parancso(ka)t hívjon. Ez a mindennapi gyakorlatban a fordítást segítő eszközt jelenti, de ezt értsük olyan általánosan, hogy az időszámításunk előtt (L.E. azaz Linux Előtt) is már implementálták kismillió platformra, pl. a Borland is DOS alá a Turbo Pascal-hoz.

Tehát összefoglalva, van egy programunk (ez lenne a make), mely futása során – összehasonlítás alapján - észleli, hogy egyes állományok frissebbek a többieknél (ezeket hívjuk függőségeknek) és ez alapján parancsokat hajt végre. Ezt a programot még rá tudjuk venni egyéb parancsok végrehajtására is illetve alapvető szöveg-behelyettesítési funkciók végrehajtására is, hogy ezen parancsokat végre tudja hajtani.

A make-ről teljes leírást a http://www.gnu.org/software/make/manual/make.html helyen találunk, értelemszerűen egy ekkora cikk csak igen vázlatos ismertetőt tud adni egy 800kB-os kézikönyvhöz képest, úgyhogy az átfogó, mélyebb ismeretekért célszerű felkeresni a fenti URL-t.

Írjunk Makefile-t!

Az egyes file-okra a függőségeket egyenként meg kell adnunk, ezeket a függőségi viszonyokat hívja a szaknyelv target-nek. A makefile alapegysége az a target, s az alábbi módon épül fel:

```
target: file függőségek
[TAB]   parancs
[TAB]   további parancs
…
…
```

Felhívjuk a figyelmet, hogy a parancsok előtt a TAB jelnek kell lennie, ha olyan szövegszerkesztővel dolgozunk, mely a TAB billentyű megnyomását átalakítja szóközökké, akkor ezt a beállítás sürgősen változtassuk meg, ugyanis a make hibát fog jelezni ha nem TAB-bal kezdődnek a parancsok. Amennyiben ha több parancsot is végre akarunk hajtatni, akkor azokat egymás alá írjuk, értelemszerűen mindegyiket TAB-bal kezdve. A target az a filenév lesz, amire a függőséget megadjuk, a függőségeknél pedig azokat a fileneveket soroljuk fel, melyektől függ a target. A parancs az bármilyen shell parancs.

```
main.o : main.c defs.h
→cc -c main.c
```

Jelen esetben a main.o fájl függ a main.c-től és a defs.h-tól, azaz ha ezen forrásfile-ok dátuma frissebb mint a main.o dátuma, úgy a cc fordítót meghívja a -c main.c paraméterekkel. Természetesen láncolhatók is a függőségek, nézzük meg rögtön egy példán:

```
edit : main.o kbd.o
→cc -o edit main.o kbd.o
main.o : main.c defs.h
→cc -c main.c
kbd.o : kbd.c defs.h
→cc -c kbd.c
```

A változók

Az eddigiekkel még nem lennénk sokkal előrébb mint a parancssor használatával, hiszen a rengeteg filenevet ugyanúgy be kell gépelni, szóval tiszta favágómunka az egész. Vagy mégsem. Az egyes neveket – elsősorban fileneveket – változókban is elhelyezhetjük, majd ezekkel a változókkal ügyes behelyettesítéseket tudunk csinálni. Legyen egy képzeletbeli programunk, ez egy szerkesztő (editor), ezért a rá való hivatkozás az edit lesz.

```
edit : main.o kbd.o command.o display.o \
       insert.o search.o files.o utils.o
        cc -o edit main.o kbd.o command.o display.o \
                   insert.o search.o files.o utils.o

main.o : main.c defs.h
        cc -c main.c
kbd.o : kbd.c defs.h command.h
```

```
        cc -c kbd.c
command.o : command.c defs.h command.h
        cc -c command.c
display.o : display.c defs.h buffer.h
        cc -c display.c
insert.o : insert.c defs.h buffer.h
        cc -c insert.c
search.o : search.c defs.h buffer.h
        cc -c search.c
files.o : files.c defs.h buffer.h command.h
        cc -c files.c
utils.o : utils.c defs.h
        cc -c utils.c
clean :
        rm edit main.o kbd.o command.o display.o \
           insert.o search.o files.o utils.o
```

Rögtön észrevehetünk egy kakukktojást, mégpedig a clean-t, mellyel a következő bekezdésben fogunk foglalkozni. Jelen esetben arra keressünk megoldást, hogy a makefile írása ne legyen ugyanolyan favágómunka mint ha parancssorból gépelnénk be, a rengeteg filenevet kellene valahogy kezelnünk. Erre ad lehetőséget a változók deklarálása:

```
 objects = main.o kbd.o command.o display.o \
          insert.o search.o files.o utils.o

edit : $(objects)
        cc -o edit $(objects)
main.o : main.c defs.h
        cc -c main.c
...
...
...
```

Létrehoztunk egy objects változót, melyre később az $(objects) módon hivatkozunk, azaz a make a $() módon leírt változó tartalmát helyettesíti be az adott helyre.

Az egyéb célok

Miután egy program fejlesztése során többféle műveletet is szoktunk végezni az állományokon, ezért egy makefile-ban több belépési pont is lehetséges, így egy makfile-lal meg tudjuk oldani az összes felmerülő műveletet. Ezeket a belépési pontokat hasonlóan definiáljuk mint a target-eket. Gyakori feladat, hogy a köztes állományokat, azaz az object fájlokat kitöröljük a forrásállományok közül, hiszen már semmi szükség nincs rájuk ha végeztünk a munkával. Ilyenkor a target nem filenév lesz hanem valami más. Nézzük például a klasszikus példát, az object file-ok kitörlését.
Az előző példában látottak szerint ilyen lehet a

```
clean :
     rm edit main.o kbd.o command.o display.o \
          insert.o search.o files.o utils.o
```

Amennyiben a make programot úgy indítjuk, hogy make clean, akkor rögtön ennél a target-nél kezdi a munkát. Természetesen a változókkal ezt is rövidebbre vehetjük:

```
clean :
        rm edit $(objects)
```

A PHONY

Szemfüles olvasó felfedezhet egy érdekes hibaforrást, mi van akkor, ha egy ilyen esetben olyan címkét adunk meg targetnek, mely mellesleg egy létező filenév? Hát bizony baleset... Ennek feloldására született meg a PHONY target, mely

használatával nem függőségként kezeli a make ezeket a címkéket.

```
.PHONY : clean
clean :
        rm edit $(objects)
```

A beépített tudás

Nézzük meg az alábbi kódrészletet és figyeljük meg, hogy mi hiányzik belőle az eddigiekhez képest, sőt úgy egyáltalán önmagában:

```
main.o : defs.h
kbd.o : defs.h command.h
command.o : defs.h command.h
display.o : defs.h buffer.h
insert.o : defs.h buffer.h
search.o : defs.h buffer.h
files.o : defs.h buffer.h command.h
utils.o : defs.h
```

Igen, maguk a parancsok hiányoznak, ám mégis működik. A megfejtés kézenfekvő, vannak a make-ben implicit szabályok, melyek közül most az érvényesül, hogy a .h illetve .c kiterjesztésű file-okból lesznek a .o kiterjesztésű file-ok és ezek fordítója a cc.

Az előre definiált változók

Az előbb említettük, hogy a C programok fordítója a cc. Mármint milyen cc? A make-ben vannak előre definiált változók, melyek használata szabványosnak tekinthető. A számunkra legfontosabbak az alábbiak, megadjuk az alapértelmezett értékeiket is:

CC : A C fordító neve. Alapértelmezése cc, emiatt a Makefile-t a CC=gcc-vel szoktuk kezdeni.
CFLAGS : A C fordításhoz szükséges flag-eket tartalmazza. Alapértelmezése üres string.
LDFLAGS : A linkeléshez szükséges flag-eket tartalmazza. Alapértelmezése üres string.
CPPFLAGS : A C preprocesszorhoz szükséges flag-eket tartalmazza. Alapértelmezése üres string.
RM : A rm -f parancs.

A változónevek és a megjegyzések

Az évek során kialakult pár konvenció a változónevekkel kapcsolatban. Az egyik, hogy jellemzőbb a nagybetűs írásmód, a másik pedig pár gyakran használt változónév elfogadottá vált, bár ezek a szabályok messze nem kötelező érvényűek.

```
## Ez a megjegyzés helye
```

A többszörös behelyettesítések

Definiáljunk egy új változót, mely az SRCS lesz:

```
SRCS = main.c kbd.c command.c display.c \
          insert.c search.c files.c utils.c

OBJS = $(SRCS:.c=.o)
```

Az OBJS változót már ennek segítségével hozzuk létre, azaz ha egy új forrásfilet csatolunk a kódhoz, úgy nem kell végigbogarásznunk az egész Makefile-t, hogy az összes helyen bővítsük az object file-ok listáját, hiszen azok automatikusan módosulnak.

Még egy érdekes lehetőség:

```
HDRS = defs.h buffer.h command.h
$(OBJS) : $(HDRS)
```

Azaz egyszerre több file-t is függőségbe hozhatunk több másik file-al, tehát ha a header file-ok közül megváltozik valamelyik, úgy az object file-okra azonnal életbe lépnek a függőségek és végrehajtódnak az újrafordítások.

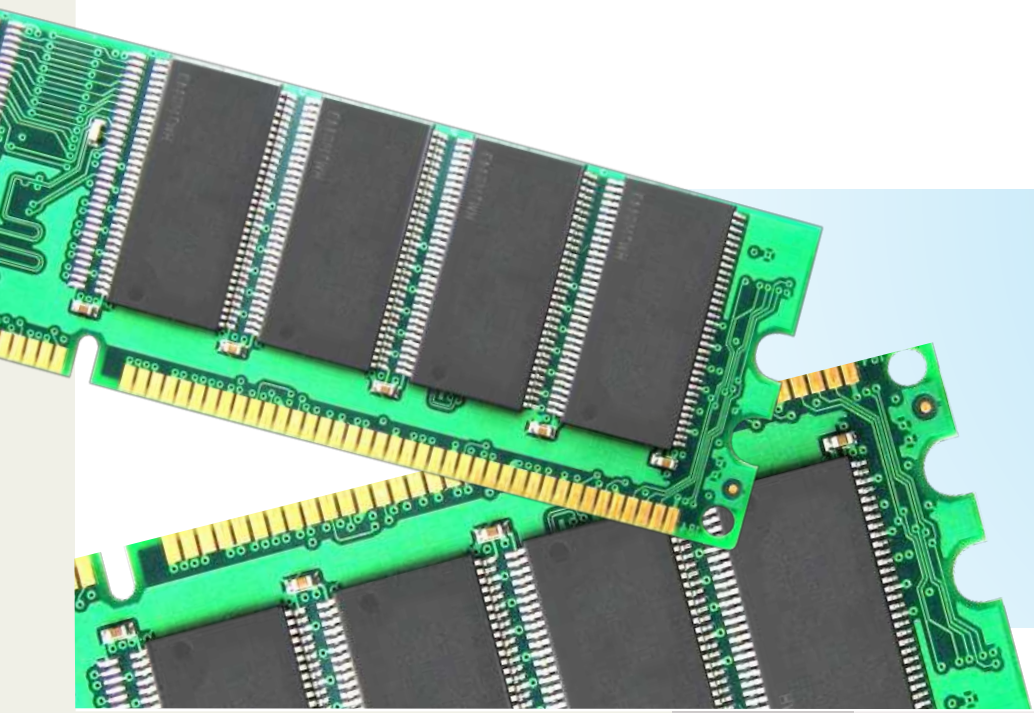
Egy példa

Az alábbi példa makefile-ban összefoglaljuk az eddigieket, majd ezek alapján házi feladat jelleggel írjunk egy olyan makefile-t, mely az eddig ismertetett programozási egységekből – main(), parancssori paraméterek bekérése, temp file-ok - összegyúr egy működő programot!

```
CC = gcc
CFLAGS = -g -I/usr/include/my_lib/
LDFLAGS = -lmy_lib

PROG = edit
HDRS = defs.h buffer.h command.h
SRCS = main.c kbd.c command.c display.c \
          insert.c search.c files.c utils.c

OBJS = $(SRCS:.c=.o)
##
##objects = main.o kbd.o command.o display.o \
##          insert.o search.o files.o utils.o
##

$(PROG) : $(OBJS)
        $(CC) $(LDFLAGS) $(OBJS) -o $(PROG)
##
##edit : $(objects)
##      cc -o edit $(objects)
##

$(OBJS) : $(HDRS)

.PHONY : clean
clean :
        $(RM) $(PROG) $(OBJS)
```

A fordítási és linkelési flag-ekkel (CFLAGS LDFLAGS) még ne foglalkozzunk, a saját Makefile-unk írásánál hagyjuk őket üresen.

Vomberg István



cracker-kánaán

Hálózatbiztonság nyílt forrású eszközökkel

2. rész

A magyar kis- és középvállalkozások hálózati védelme súlyos biztonsági és működtetési hiányosságokkal segíti a rosszindulatú behatolókat. Hálózatbiztonsági sorozatunk mostani fejezetében ezeket a hibákat tekintjük át.

Az első rész végén valamilyen támadást ígértünk (legyen ez most vírus), de a roham előtt egy-két dologgal még foglalkoznunk kell. Nézzük meg először is az alábbi tanulságos listát. Itt különböző felmérésekre alapozott becsléseket találunk arról, hogy a magyar vállalatok, vállalkozások, hogyan védekeznek a különböző fenyegetettségek ellen. A néhány fős minivállalatok és egyéni vállalkozók adatai nem szerepelnek, különben a helyzet még siralmasabb lenne. A legtöbb cég még az alapvető biztonsági fenyegetettségek ellen sem védekezik megfelelően.

Vírusvédelmi rendszereket, programokat 90–95%-uk használ, ez első ránézésre jónak tűnik, de mégsem az, mint később látni fogjuk. Rosszabb a helyzet, ha a tűzfalmegoldásokat nézzük. Itt 70–75%-os arányt találunk, ami valószínűleg a számadat által sugalltnál jóval gyengébb lehet, ha figyelembe vesszük, hogy nem elég beszerezni egy tűzfalat, hanem azt üzemeltetni is kell és a tűzfal megoldások sem egyformák. A levélszemét szűrését már csak a cégek mintegy fele végzi, pedig érdemes belegondolni, miféle fura jószágok juthatnak be rendszerünkbe a levelezés mögé bújva. Jelszó alapú azonosítás kevesebb, mint 50%-nál található, ennek egy része valószínűleg kimerül a monitorra ragasztott, vagy terítőre hímzett 3 karakteres bonyolult kifejezések használatában, mint mondjuk az ági, vagy az ili. Ja, elfelejtettem az icát, így csupa kisbetűvel ica, ica (felhasználónév, jelszó) saját tapasztalat. El kell ismerni, hogy ezzel máris „van” az adott helyen jelszó alapú azonosítás. A spyware, adware, és hasonlók szűrésével kapcsolatban jó hír, hogy létezik ilyen, a rossz hír, hogy csupán 30% körüli az arány. Pedig itt találhatnánk igazi csemegéket.

Nagyon fontos témakör nálunk is, de a fejlettebb országokban is – mint például az USA – a fájlokhoz való hozzáférés kontrollja. Annyira fontos, hogy egyes szakértők szerint nincs is szükség a határvédelemre, a legfontosabb dolog az állományokhoz való hozzáférés szabályozása, az állományvédelem. Magyarországon azonban biztos, hogy még nem felejthető el a határvédelem sem, bármennyire is egyet kell értenünk az állományhozzáférési kontroll fontosságával. A következő adat is ezt bizonyítja: nálunk céges környezetben a fájlokhoz való hozzáférést a vállalkozások kevesebb 20–25%-a korlátozza. Néhány helyen szívesen megnézném, hogyan megy ez a gyakorlatban, de nem feltételezek túl sokat.

A hálózati hozzáférés kontrollja még ritkább, pedig ezzel elég jól meg lehetne fogni a dolgokat. Ez körülbelül 15%-ra tehető. IT szabályzat is nagyjából ennyi vállalkozásnál lehet, talán egy-két százalékkal kevesebbnél. Ami nevetséges, hogy ezek közül nagyon sok helyen figyelmen kívül hagyják a meglévő IT szabályzatot. Megszegése miatt semmiféle szankcióra nem kell számítani, sőt, nagyon sok helyen adott formájában betarthatatlan. Így nem csoda, hogy ez nem érdekel senkit a szerencsétlen egyetlen informatikus (szerelő, lakatos, …) rabszolgán kívül, vagy még őt sem, csak a papírokon az ő neve szerepel.

A következő a listában az elektronikus aláírás, ezzel a cégek 10-11%-a próbál élni. Ugyanennyire tehető a behatolásérzékelés és -védelem (IDS), de ezzel sem mennek sokra az olyan helyeken, ahol a riasztásokra nincs, aki reagáljon és még a logolvasás is a felejtős tevékenységek közé tartozik. A többi megoldás nem számottevő, nem ismerik, nem alkalmazzák őket megfelelő arányban.

Ami ebből az egészből kiderül, az az, hogy Magyarország nemcsak szexparadicsom, de cracker-kánaán is egyben, csak legyenek olyanok, akiket érdekelnek az itteni adatok, információk. A felsorolásból leszűrhető, hogy a védekezésre többféle lehetőségünk is van és ezeket kombinálhatjuk is, bár nem sokat érünk a védelmi rendszerek halmozásával, ha ezeket nem tudjuk megfelelően konfigurálni és üzemeltetni - amint ezt egyébként sok cég megteszi.

Akkor most jöjjön a lista első helyén álló vírus témakör, abból is a támadás és a védekezés kérdései. Magyarország különleges ország. Míg szinte az összes demokratikusnak mondott helyen, széles irodalma van a hálózatbiztonsági kérdéskörnek, beleértve a támadások konkrét leírását is, addig nálunk az ethical hacking ezen része tabu, mivel a hosszú börtönévekkel jutalmazott bűncselekmények kategóriájába esik. Érdemes az egész paragrafust eredetiben is áttanulmányozni, de ami leszűrhető röviden, az az hogy már az adatok, módszerek nyilvánosságra hozatalával bűncselekményt követünk el, függetlenül attól, hogy valaki ezt felhasználja-e a felsorolt bűncselekmények elkövetésére. A másik probléma pedig az, hogy nincs lehetőségünk autentikálni azokat a személyeket, akik a magazint elolvassák. Ezért azt a megoldást fogjuk alkalmazni, hogy megemlítjük a támadási lehetőségeket, módokat, valamint az elkövetők ismertebb eszközeit. Részletesen nem írhatjuk le magát a támadást, de a védekezési lehetőségeket igen.

A vírusfajták felsorolása helyett általánosságban megállapíthatjuk, hogy mind ezek, mind a készítők motivációja jelentősen megváltozott az évek során. A kezdeti kihívás szintű, majd rombolási szándékú aktivitást felváltotta a hivatásszerű pénzszerzést célzó, minél nagyobb extraprofitot elérni kívánó mentalitás. Ennek eredménye a világszerte megfigyelhető hatalmas zombihálózatok létrejötte. Ezek komoly pénztermelő „üzemek”, ezért nem is cél a gépek tönkretétele, a szolgáltatások teljes megsemmisítése, csak a megszállásuk, a felügyelet bizonyos szintű átvétele, az irányítás megszerzése. Fontos a feltűnés lehető legnagyobb mértékű kerülése. (Ez ismerős lehet a kiváló sci-fi filmekből is, ahol a gazdaszervezetet kis élősködők irányítják, mindaddig amíg arra szükségük van. Ráadásul legtöbbször hosszú ideig a gép üzemeltetőjének fogalma sincs arról, mi zajlik a háttérben). A támadás mindenfelől és minden módon bekövetkezhet, érdemes a ma terjengő kártevőket áttanulmányozni, van itt valódi vírustól, a trójain keresztül a wormig minden. Támadnak a hálózat irányából, a különböző mágneses és optikai lemezeken, és az USB-n keresztül is. A hordozható készülékek csatlakoztatása is veszélyt jelent, hiszen ma már a kellően fejlett mobilokon is vannak vírusok.

Mindebből látszik, hogy a szerveroldali vírusvédelmet ki kell egészíteni a munkaállomásokon futó vírusvédelmi rendszerrel is, nem elég csak a vírusok ellen védekezni, a többi kártevőt is komolyan kell venni. Van-e értelme egyszerre több víruskeresőt futtatni? Nem nagyon, mivel azon felül, hogy jelentősen lelassítják a rendszer működését, a memóriában ügyködő moduljaik általában összeférhetetlenek. Ennél jobb módszer egy általunk megbízhatónak tartott keresőt alkalmazni, és időről időre, különösen gyanús viselkedés esetén, másik rendszerrel is leellenőrizni a gépet. Nyilván a vakriasztások száma a használt rendszerek arányában nőni fog. A vakriasztások és a jelentős számban terjedő, idióta hoaxok miatt még ezen az egyszerűnek tűnő (szinte teljesen automatikusan működő, saját magát rendszeresen frissítő) védelmi területen sem szabad magára hagyni az átlagfelhasználót. Sok esetben ugyanis képtelen elkülöníteni a vakriasztást a valódi fenyegetéstől, és igen erős hajlama van az albán vírus jellegű utasítások gépies végrehajtására is.

Objektum:
http://www.faqs.org/qa/qa-13492.html
Kártevő:
valószínűleg ismeretlen SCRIPT vírus
Információk:
kapcsolat megszakítva - karanténba helyezve

A vírusvédelmi programokról már elég sok ember hallott, de az már nem annyira köztudott, hogy a nyílt forrású, avagy szabad alkalmazásoknak ezen a területen sincs okuk a szégyenkezésre. Ingyenes antivírus programokkal akár Dunát is lehetne rekeszteni, főleg a legelterjedtebb – nem szabad – operációs rendszerre írottakkal, de van nyílt forráskódú alkalmazás is a legtöbb platformra. Ezek egyik méltán ismert képviselője a Clam Antivírus és a hozzá kapcsolódó oldalkocsik (ClamMail, SquidClamav, stb). Unix/Linux desktopokon is létez(het)nek vírusok, de jelentőségük nem olyan nagy, mint más rendszereken. (Ezt a tényt vegyük tudomásul és kerüljük a parttalan vitákat arról, hogy ez miért van így). A kártevőkkel kapcsolatos másik fontos védelmi lehetőség mind nyílt, mind zárt környezetben, hogy kerüljük a rendszer teljes jogú felhasználóként való használatát. Ilyet csak az igazán nélkülözhetetlen esetekben tegyünk. Ugyanez vonatkozik a futó szolgáltatásokra is, csak a legszükségesebbek fussanak, a működésükhöz feltétlenül szükséges, minimális jogosultságokkal. Míg a nyílt rendszerek figyelnek és figyelmeztetnek arra, hogy ne legyünk mindig „gyökerek”, addig a Windows legtöbb verziója könnyen elfogadja alapbeállításként a rendszeradminisztrátor állandó, és jelszót nélkülöző használatát is. Az alapbeállításokat pedig a legtöbb felhasználó nem fogja megváltoztatni, és ezt a cracker bajtársak is jól tudják. Még egy fontos szempont: ha úgy döntünk, hogy az interneten keresünk víruskészítéssel kapcsolatos információkat, vagy vírusgenerátor alkalmazásokat, csak jól védett gépről tegyük, hiszen amint az első oldalak valamelyikére tévedünk, máris jó eséllyel megpróbálnak elhelyezni a gépünkön valamilyen kedves alkalmazást. Aki bízik magában és a gépe védelmi rendszerében, kipróbálhatja (persze a Windows-gépek megint előnyben).

A média kedvencei (és a script kiddie-ké is) az úgynevezett vírusgenerátorok, laborok. Ezzel kapcsolatos jó hír, hogy a „forgalomba került” víruslaborok csak a legelső időkben okozhatnak problémát, egy jól karbantartott rendszerben, hiszen a víruskereső programok gyártóinak elemi érdeke a gyors reagálás, így aztán hamar védettséget biztosítanak az egyes vírusgyárak termékeivel szemben. Vagyis a hozzánk került vírusgyárral elkészített vírus jó eséllyel fennakad a frissített vírusvédelmi program szűrőjén, de egészen más a helyzet, ha valóban tudunk készíteni egy egyedi vírust. Ehhez viszont már valódi szaktudás kell. Tesztelhetjük rendszereinket bármelyik ismert vírusgenerátor által készített jószágokkal is. Ilyenkor legyünk tudatában, hogy a saját felelősségünkre tevékenykedünk, annak minden következményével együtt. Védekezés a már említett módokon lehetséges, soha se feledjük a legfontosabbat, hogy mindig naprakész, friss rendszert használjunk (a víruskeresőt is beleértve). És nem árt, ha megbízható, ismert vírusirtót használunk, mivel vannak álvírusirtók is, ne dőljünk be nekik.

A legfontosabb protokollok ismerete nélkül vajmi keveset tehetünk a hálózatbiztonság területén, ezért sorozatunk következő részében ismét elővesszük azokat.

Szőke József

http://www.kissgaborzsolt.hu/dokumentumok/informatikai-incidens/2001.-evi-cxxi.-torveny.html

Impresszum

Alapító-főszerkesztő:
Horváth Örs Apor

Felelős szerkesztők:
Jankovich Oszkár
Pfeiffer Szilárd

Tördelőszerkesztő:
Barta Károly

Arculattervező:
Makay József

Szerzők:
Csíkos Bálint
Kovács Zsolt
Medve Zoltán
Sütő János
Szőke József
Vomberg István

A II. évfolyam 2. szám fórumának címe:
http://flosszine.org/forum/II_evfolyam_2_szam

A FLOSSzine elérhetőségei:
E-mail: info@flosszine.org
Web: www.FLOSSzine.org
IRC: #FLOSSzine ; #FLOSSzine.hu ; #FLOSSzine.org (irc.freenode.net)

Köszönet az FSF.hu Alapitványnak a tárhelyért!

Az e-fanzine elkészítéséhez kizárólag nyílt forráskódú, szabad és ingyenes szoftvereket használunk.