FLOSS ZINE

Free / Libre / Open Source Software fanzine

2008. október

I. évfolyam 2. szám

Hogyan kezdődött?

Ubuntu konferencia 2008 beszámoló

sylsog-ng 3.0

FOLYTASSUK A PROGRAMOZÁST!

LITE:

Hogyan kezdődött?
Hacker-portrék: Eric S. Raymond
Ubuntu@hu
Warsaw
INTERVIEW: Scheidler Balázs (syslog-ng)

PRO:

Bash alapok
Az indián nyomában (LIGHTTPD)
Webmin
Mi van ott? (amap, vmap)

LITE

PRO

Hogyan kezdődött?

Kissé nehéz helyzetben van, aki olyan időszakokról akar írni, amelyeknek nem volt részese. Mivel azért ez mindennapos dolog, én is megpróbálkozom vele. A számítástechnika kezdeteiről (is) írni, olyasmi, mintha 56-ról írnék, (egyiknél sem voltam még meg).

A fanzine neve FLOSSzine, ezért jó lenne a nyílt forrású dolgokról írni, de azok sem csak a semmiből materializálódtak (ez szoftver esetén egyébként is érdekes dolog), így érdemes az előzményekre is szánni pár szót. A nyílt forrású rendszereket megelőző időkkel kezdeném, egészen a korai időkre visszatekintve, és talán egy kis vitát generálva (annak ellenére, hogy mindenre van definíció), általánosságban megnézve a szoftverek, programok mibenlétét. Ha a számítástechnikán túl tekintünk, lehet hardvernek nevezni a baltát? Akkor az azt működtető kéz a szoftver, (persze egy ökölcsapás esetén simán hardvernek minősül), vagy a gondolat, amely meghatározza kit, vagy mit csapjon ketté a balta?

A számítástechnika területén sokkal könnyebb helyzetben vagyunk. Világos definíciók vannak ezekre a dolgokra (persze itt is vannak vitatható területek, tételek, pl.: maga az információ).

Mi is az a szoftver (persze, ami miatt belerúgunk a hardverbe), vajon lehet-e ennél meghatóbban körülírni a dolgot?

Volt-e mindig szoftver? A mai formájában biztosan nem.

A számoló, és a számítógépeket pontosan definiálhatjuk (http://hu.wikipedia.org/wiki/Számítógép).

Tágabb értelemben a mai számítógépek őseinek a különböző számolást elősegítő eszközöket lehet nevezni – ilyen eszköz az ókori eredetű abakusz, vagy a számítástechnika korai szakaszában, a 3100 éves (körülbelül) inka kipu és yupana.

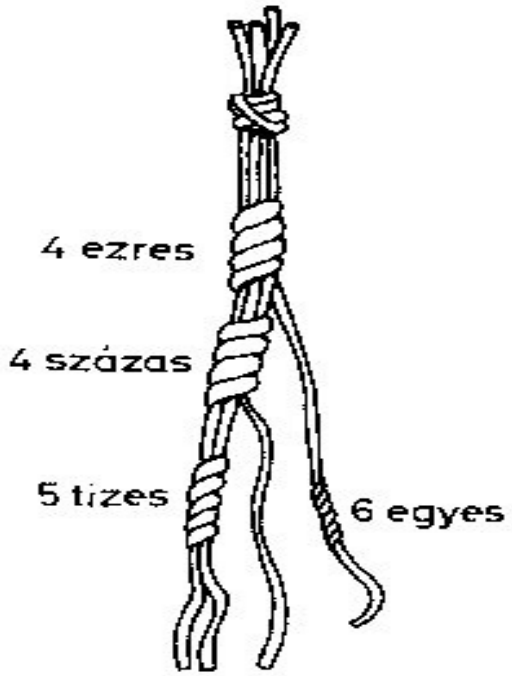
A kipu a kecsuák nyelvén csomót jelent, használati gyakorlatáról Leland L. Locke tudománytörténész már 1923-ban bebizonyította, hogy nem pusztán dekoratív elem, hanem afféle textil abakusz, ahol a csomók jelentést hordoznak.

Az inka birodalom i.e. 1100 körül alakult ki az Andok magasföldjein, és a 16. századi spanyol hódításig állt fenn, de sohasem volt írásrendszere, nem maradt fenn írott nyelvemléke. Az inkák 1200 körül a világ legnagyobb államát tartották fenn. De hogyan volt ez lehetséges írásrendszer nélkül?

A Harvard Egyetem antropológusa, Gary Urton könyve (Signs of the Inka Khipu) szerint az inkák a kipuk hétbites bináris kódrendszere segítségével irányították birodalmukat.

A kipu készítése közben hét ponton rendre két lehetőség között választott a készítő.

A kipu anyaga (gyapot vagy gyapjú), a fonál sodrásiránya, a csomózás iránya, a szálsűrű-



ség, és egyéb tulajdonságok alapján összesen 128 permutáció (kettő a hetediken) jön ki, mely a 24 különböző szín használatának lehetőségével szorozva már 1536 információegységet jelent.
A sumérok 1000-1500 ékírásos szimbólumához képest ez feltétlenül jelentős többlet, az egyiptomi illetve Maja hieroglifekhez képest pedig kétszer több.
Az inkák egy, a Fibonacci-számokon alapuló számolási rendszert is kifejlesztettek, amellyel a legkisebb hiba nélkül lehet rendkívül bonyolult kalkulációkat végezni. Ezt yupana számolótáblák segítségével alkalmazták, amelyeket a nálunk is ismert abakuszhoz lehet hasonlítani.
Ebben az esetben a kipu tekinthető adathordozónak, a yupana pedig az adott kor számológépének.
(A kipunál az anyaga a hardver, vagy inkább a yupana (így a kipu inkább adathordozó, mint pl.:a mágneslemez)), a kipu színe, csomói, stb. a szoftver (a rajta lévő adatok, algoritmusok)? (Mivel a spanyolok nem értették a csomóírást, összeesküvéstől tartva minden ilyen eszközt megsemmisítettek, ezért nagyon kevés kipu és yupana maradt fenn, a használatukról pedig semmi konkrétumot nem tudunk).
Később a dugaszok, lyukkártyák, lyukszalagok voltak az adat-, és szoftverhordozók. A lyukkártyákat nem lyukasztották át újra, nem módosították a rajta lévő információt. (Szigorúan véve a szoftver is hardver volt régebben?)
Ezek kiváló kérdések, a flame labda fel van dobva, jöhetnek a vélemények, főleg szoftver-hardver témakörben, de abban azt hiszem megegyezhetünk, hogy ezekben az esetekben (és még sokkal később is), a szoftver ingyenes volt és szabad, sőt az említett esetekben része az egész rendszernek.
Persze a működőképesség tekintetében, ma is rengeteg szoftver része a számítógépnek, ami nélkül nem működne, legalábbis számunkra hasznosan nem (pl.: BIOS).

Egyébként a lyukkártyás, lyukszalagos megoldások is sokkal régebbiek, mint azt sokan gondolják, már a mechanikus számológépek korában használták őket.
Van itt egy szintén érdekes megoldás, 1275 (körül) Raymundus Lullus feltalálta logikai gépét. Az „Ars Combinatoria” című munkájában írta le az első, általunk ismert szöveg-gépet, amely sajátos mechanikus módszerével képes volt igaz (és hamis) állításokat produkálni. Ennél a masinánál a cserélhető papírkorongok jelenthetik a szoftvert. Ez a gép nem a számolási feladatok megkönnyítése céljából jött létre, (de ebbe mélyebben nem lenne szerencsés itt belemenni), mégis többek szerint a Turing gép ősének tekinthető, Werner Künzel szerint Lullus olyan logikai gépet talált föl, amely eredményeket, állításokat produkál - általában véve output adatokat - egy világosan meghatározott mechanikus algoritmus segítségével!
A leírást később nagyon sokan használták kiindulásként saját munkájukhoz.

Szintén mérföldkőnek nevezhető Jacquard automata szövőgépe.
Folyamatok vezérlésére már évszázadok óta alkalmaztak különböző vezérlési módokat. Zenegépekben pl. a tüskés henger volt a jellemző megoldás. A henger mérete (vagyis a kerülete, mivel azon voltak a tüskék) természetesen megszabta a program hosszát: a henger minden körülfordulása ugyanazt a tevékenységet idézte elő. A mintás szövés vezérlésére viszont olyan módszer kellett, amivel egyrészt hosszabb programot is meg lehet adni, másrészt pedig viszonylag egyszerűen lehet a mintát megváltoztatni, a szövő-

széket “átprogramozni”. Az idők folyamán többféle ilyen vezérlést találtak fel. Brösel 1690 körül vászonszalagra faelemeket ragasztott, ezzel határozták meg a szőtt anyag mintáját. A mintát a vászonszalagok cseréjével lehetett változtatni. A lyoni selyemszövőgépekben kb. 1725 óta lyukasztott papírcsíkok látták el ugyanezt a feladatot. Joseph Marie Jacquard (1752-1834) francia feltaláló a vezérlést tovább tökéletesítette. 1810-ben (1804-ben?) olyan automatikus szövőszéket tervezett, amelynél fából készült vékony, megfelelően kilyuggatott lapok (“kártyák”) vezérelték a bonyolult minták szövését. A lyukkártyákat láncra fűzte, ezzel lehetővé téve a minták (azaz a szövőszék vezérlésének) gyors és könnyű megváltoztatását. (Ez a “gyors és könnyű” állítólag mintegy 15 napos munkát jelentett.)

Érdemes megemlíteni még a Hollerith-féle lyukkártyás adatfeldolgozást, amit 1889-ben szabadalmaztatott, majd ezzel a módszerrel dolgozta fel az USA 1890-es népszámlálási adatait, mindössze 4 hét alatt, (a korábbi feldolgozás 7 évig tartott). Ennek sikere nyomán, 1896-ban megalapította a Tabulating Machine Company nevű céget, amelyből aztán 1924-ben létrejött az IBM.

Mindezekből is látható, hogy az úgynevezett informatikai forradalom semmiképpen sem volt előzmények nélküli (ráadásul nagyon kevés dolgot hoztunk fel példának, a régi szerkezetek, automaták leírásával több könyvet meg lehetne tölteni, akkor is, ha csak a fennmaradt, működő és rekonstruálható berendezésekre gondolunk).

Nézzünk sokkal későbbi dolgokat, de előbb tisztázzuk, a ma leggyakrabban számítógépnek nevezett berendezések elektronikus, digitális számítógépek (ezért nem beszélek a mégoly érdekes mechanikus automatákról, vagy elektronikus, de analóg számítógépekről sem a későbbiekben).
Fentiek a Neumann János által néha automatáknak is nevezett gépek. A későbbiek folyamán elterjedt, számítógépnek nevezett eszközök a Neumann-elveknek megfelelően megalkotott számítógépek. Vagyis (szűkebben) a gép módosítható memóriájában helyezkedik el a szekvenciális szervezésű program, amely meghatározza a központi egység által végrehajtott alapvető számítási lépések természetét és sorrendjét. A memória és az adathordozók egyaránt tárolhatnak adatokat és programkódot.
Ma a mindennapi használatban főleg ilyen gépeket használunk, ugyanakkor agyunk olyan számítógép (hardver), aminek a szoftveréről nemigen tudunk semmit.

Az első elektronikus, digitális számoló és számítógépek programozása, dugaszolással, huzalozással történt.
A gép átdugaszolása sokáig tartott. Később a lyukkártya, lyukszalag idejében is sokszor tovább tartott a program „lyukasztása”, mint maga a számítási műveletsor.
Látható, hogy ebben az időben nem vált el a szoftver és a hardver (annyira nem, hogy maga a szoftver kifejezés is csak 1958-ban keletkezett, ami John W. Tukey alkotása).
Magának a programnak az elkészítése is csak a számítási feladatsor megfogalmazása után, a helyszínen, vagy annak közelében történt.
Fontos tényező az is, hogy a feladat megoldásának ideje alatt a kapcsolatminta (a dugaszok helyzete, a lyukkártya vagy szalag), vagyis a program nem változott.
Ekkor a programnak nem jelentkezett külön ára, bár az biztos, hogy nem ingyen állították elő.

Néhány gép ebből az időszakból:

ENIAC

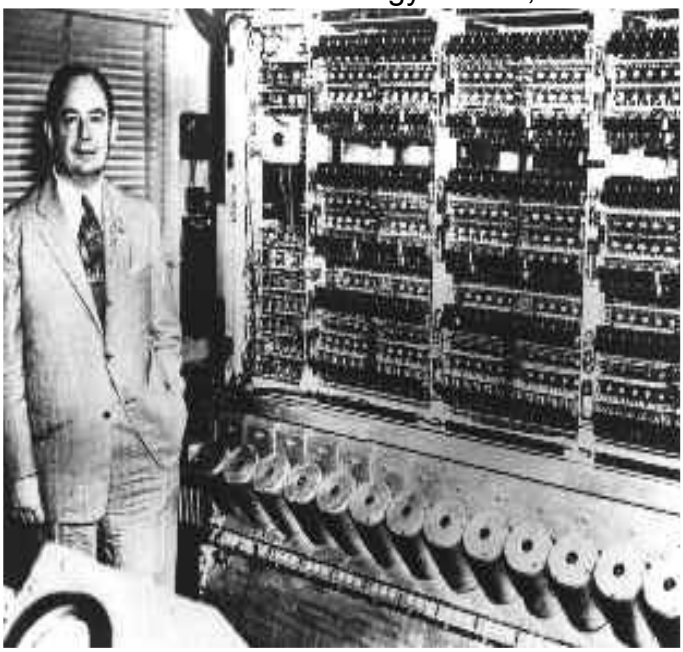
1946-ban készült el az ENIAC (Electronic Numerical Integrator And Computer), ami hivatalosan az első programozható, elektronikus, digitális számítógép volt.
Tízes számrendszerben működött, tízjegyű előjeles számokat kezelt - aritmetikai egységei több feladatot is elvégeztek egyszerre. Az elektroncsöves flip-flopokból összeállított regisztereibe impulzussorozatokkal vitték be a kívánt számokat, és az állandókat kapcsolókkal állították be. A programot lyukkártyákra lyukasztották, és az adatokat 20 db tízjegyű regiszterben tárolták.

EDVAC

1949-ben megjelent az EDVAC (Electronic Discrete Variable Computer), amely Neumann János elvei alapján, az ő közreműködésével készült. Ez volt az első, belső programvezérlésű, elektronikus, digitális, univerzális számítógép.
Az EDVAC sok fontos vonásban különbözött elődeitől. Sokkal nagyobb memóriakapacitása volt: egy elsődleges, 1024 szavas higanykésleltetővonalas operatív tár és egy másodlagos, lassabb, mintegy 20 kilószó kapacitású mágnesdrótos tár. Mivel a késleltetővonalas tár soros (bitenkénti) elérésű volt, ezért az aritmetikai-logikai egység is soros volt, bitenként dolgozta fel az adatokat. A gép négycímes utasításokat használt: aritmetikai utasításoknál ebből kettő volt a két operandus címe, egy az eredmény címe, és egy a következőként végrehajtandó utasítás címe. Egy program végrehajtásához előbb az egész programot és az adatokat be kellett táplálni a memóriába. Adatbevitelre egy írógépszerű eszközt használtak, ami közvetlenül a mágnesdrótra írta az információt. Adatkivitelre egy nyomtatót alkalmaztak.

Ettől kezdve már a papírból készült lyukszalag olvasási sebessége nem korlátozta a számítógép sebességét, és egy új probléma megoldásához nem kellett a gépet áthuzalozni. Több mint 85 négyzetméteres szerkezet volt.

Az ezt követő időszakban megjelentek a sorozatban gyártott számítógépek, és a számítástechnika egyre inkább hozzáférhetővé vált. (Ebben ismét nagy szerepe van Neumann Jánosnak, aki ragaszkodott hozzá, hogy a számítógép elvi leírását megosszák másokkal, a „First Draft of a Report on the Edvac” című művében (eredetileg ezt csak munkaanyagnak szánta, de nagyon sok másolat készült róla) leírta azokat az alapelveket, amelyeket ma a „Neumann-elvek”-nek nevezünk. Sőt ragaszkodott ahhoz is, hogy a következő IAS gép (ez már párhuzamos szervezésű volt) leírását is nyilvánosságra hozzák, így ezután évekig ezeket a gépeket másolták a világ más pontjain (EDVAC->EDSAK, BESK, DASK), (IAS->ILLIAC, ORDVAC, MANIAC, JOHNNYAC, MESM, BESM, M-3, IBM). Neumann mindig törekedett rá, hogy a tervezési alapelvek, és a berendezések működési és funkcionális jellemzői nyilvánosságra kerüljenek. A számítógépek elkészítése (másolása) ezen adatok ismeretében sokkal könnyebbé vált. (A First Draft...-ot, sok más országhoz hasonlóan a szovjetek is elkérték, és H. H. Goldstein szerint meg is kapták. Neumann és Goldstein nagyon sok anyagát, jelentését elkérték mások. Valószínűleg ennek (és a többi hozzáférhető jelentésnek) az alapján készült az előbb is említett MESzM, a BESzM és még nagyon sok számítógép az ország különböző részein).
Ennek tükrében bátran nevezhetjük Neumann Jánost a nyílt forrású eszmék egyik (számítástechnikai viszonylatban talán legelső) zászlóvivőjének.
A kezdeti időszakban az egyetemek, kutatóin-



tézetek programfejlesztéseiket megosztották egymás között, a beszerzett számítógépekhez hozzátartoztak az alapvető programok, a hiányzókat kifejlesztették, így a legtöbb esetben nem merült fel (mai formájában) a szoftverek árának kérdése.

Ebben az időszakban a számítógépeket nem csupaszon árulták, hanem a működésükhöz szükséges szoftverekkel együtt.
Később vált külön, amikor a (csak) szoftverértékesítésből élő cégek, zárt forrású alkalmazásokkal rukkoltak elő, sokszor szinte a semmiből. Ehhez mindenképpen kellett a számítástechnikai berendezések nagyfokú elterjedése.
Ettől függetlenül az ingyenes szoftver folyamatosan jelen volt (shareware, public domain, freeware, cardware, stb. formájában). Majd újra visszatért gyökereihez, és talán egyszer ismét szinte kizárólagos lesz, és a nyílt, ingyenes szoftver kiszorítja a kereskedelmi termékeket.
A kétféle megközelítés, nyílt és kereskedelmi között mindig is volt átjárás. A nyílt közösség sok kereskedelmi termékhez hozzátett, de a kereskedelmi cégek is (főleg mostanában) sok, valóban fontos lépést tettek a nyílt rendszerek elterjedésének érdekében (ne legyenek illúzióink, ezt is keményen üzleti megfontolásból tették).

A mai értelemben vett szabad (free) szoftverek megjelenése (és értelmezése), a GNU kiáltványhoz és a GPL-hez köthető. (GNU 1984, GPL 1988-89).

Ebben az időszakban Magyarországot még sújtotta a COCOM lista. Hazánk vonatkozásában a COCOM listát 1990-ben enyhítették, 1992. február 10-én pedig véglegesen megszüntették, miután bevezettük a COCOM által szükségesnek vélt exportkorlátozásokat.

Mégis mi volt Magyarországon a 80-as években? (Inkább csak néhány, elterjedtebb gépet említek).
Saját fejlesztésű vagy ESZR (TPA, R10, HT 1080Z, Primo, Videoton TV Computer), szocialista importból származó (pl.: Robotron), és nagyon sokszor magánimportból származó (éveken keresztül túlnyomórészt COCOM listás) gépek.
Nagyon sok berendezést a magyar vámtörvények miatt is, de a COCOM tiltás miatt is szétszerelt állapotban kellett behozni.
Egy Epson FX-1000 mátrixnyomtató minimum két részből állt (legalábbis papíron), nyomtató mechanika és nyomtató elektronika néven futott az egész (szerencsés esetben nem kellett valóban szétszedni, mert nem ellenőrizték, de ez inkább az időszak vége felé volt jellemző).
Az IBM PC XT és AT gépekkel is hasonló volt a helyzet.
A magyar vállalatoknál hétköznap Commodore 64-es gépeken „komoly” ügyviteli programokat futtattak, hétvégén játszottak velük (na, jó, néha hétközben is).
Szoftverek? „Ingyenesek” többnyire, ritkán méregdrága, de csak a legritkább esetekben legális programokkal volt tele minden.
Másolt mindenki, mint a güzü, ha tehette, komoly cserebere alakult ki, sokan a programgyűjteményük tizedét sem látták soha működés közben.
Ebben az időszakban elterjedt gépek voltak még az alábbiak. (Az eddig felsoroltakon kívül).

Commodore VIC-20
Commodore 64 (a 80-as évek közepéig COCOM listán, de ez a behozatalért felelős magánhadseregünket nem nagyon zavarta).
Commodore C16
Commodore Plus/4
Commodore 128
Commodore Amiga.
A Sinclair ZX Spectrum különböző fajtái és a ZX80, ZX81.
ZX Spectrum (1982), ZX Spectrum+ (1984), ZX Spectrum 128K (1986), ZX Spectrum +2 (1986),
ZX Spectrum +3 (1987), ZX Spectrum +2A / +2B (1987).
ATARI gépek (főleg 600, 800XL, 520ST és 1040ST).
Enterprise 128
Amstrad – Schneider CPC sorozat
TI-99

Ezeknek a gépeknek a nagy része magánimportból került az országba, bár egyes gépeket hivatalosan is forgalmaztak, néha nagyobb mennyiségben is (Plus/4, Enterprise).

Nagyobb gépek VAX, MicroVAX, PDP, TPA, stb.
Ezen felül biztosan nagyon sokféle gép létezett már akkoriban is Magyarországon, de megpróbáltam a leginkább elterjedtekre és nagy számban létezőkre szorítkozni.

Sokan titulálták játékgépnek a C64-et, és ebben teljesen igazuk is van. Ennek ellenére itthon nagyon sok cég abban az időszakban minden számítógépet igénylő feladatot ilyen gépeken végzett el.
Az igazsághoz az is hozzátartozik, hogy az egyik elterjedt Robotron masina, az 5110-es nagyon drága volt, 64 KB memóriával rendelkezett, és még csak rendes játékok sem voltak rá. Cserébe viszont kiválóan tudott kartonozni.
Az 5110-es egyik legizgalmasabb alkalmazása egy zöld monitoron ugráló karakter volt (más karaktereket kellett vele átugratni), amit lóversenynek hívtak. Az adatrögzítő lányok mindenkinek eldicsekedtek vele (szegényeknek akkor még nem jutott egy normális pasziánsz sem), de hát aki férfiember akkoriban abba a szobába tévedt, annak mindnek volt otthon egy-két mikroszámítógépe, Commodore, Spektrum, Atari, stb, amik már akkor színesebbek, szagosabbak voltak a Robotronoknál, (mondjuk kartonozni egyik sem tudott) így aztán csak bambán meredtek az NDK-s csoda zöld képernyőjén pattogó micsodákra, és amikor megtudták, hogy a fél szobányi berendezés 64 KB memóriával bír, rögvest kifordultak az ajtón.

A szoftvereket ekkoriban cserélték, másolták, és ha valaki szörnyen perverz módon teljesen

hivatalossá, legálissá szerette volna tenni szoftverbeszerzését, nagy nehézségekbe ütközött.
Cocom lista, vagyis hivatalosan nem is lehetett beszerezni nagyon sok terméket, az ekkoriban az országba került szoftverek dobozain legtöbbször ott virított a „csak az USA-ban forgalmazható” felirat. Nézzük a másik oldalát a dolognak (az áram is a kisebb ellenállás irányába halad), minek kínlódna valaki drága és bonyolult szoftverbeszerzési folyamatokkal, amikor sokkal könnyebben megoldható a dolog. Ebben az időben a BSA még csak a répában volt vitamin (a BSA-t 1988-ban alapították, Magyarországon 1994-ben kezdte meg áldásos tevékenységét), így nem nagyon ijesztgethette a jámbor számítógép felhasználókat, nem nagyon volt, ami rászoríthatta volna az embereket a legális szoftverhasználatra, nem voltak hivatalos beszerzési források sem.
A hardver iszonyú drága volt, 1987-ben egy (szinte legkisebb) Novell hálózat (1 db AT, 2 db XT gép, Arc-Net csatolókártyákkal, kábelekkel) eszközeinek ára 1,2-1,7 M Ft volt, ezek után nem nagyon volt kedve senkinek százezreket, milliókat költenie szoftverre, de nem is nagyon volt honnan.
Mivel a hálózatnak működnie kellett, automatikusan járt hozzá a hálózati program, másolt lemezekről. Ekkor kérdeztük meg, vajon hivatalos, legális-e a Novell? Először bambán néztek ránk, hogy mi bajunk van, majd valahonnan előkapartak néhány piros címkéjű lemezt, meg egy-két piros dobozt, és azt mondták itt van. Licence szerződés, ilyesmi persze nem volt.
A hálózat működött, a gépeket kiszolgáló Netware 286 2.x még valószínűleg ma is tenné a dolgát, ha a gépek ki nem pusztultak volna alóla (ne felejtsük el, akkoriban internet nem volt még).

Egy bekezdés erejéig térjünk vissza az úgynevezett játékgépre, a Commodore 64-re. Ehhez egy időben már adtak (mágneslemezen) egy Geos nevezetű kiváló rendszert, ami akkoriban valóban csodaszámba ment. (Mivel a géphez adták, nevezhetjük ingyenes programnak). Mindenesetre nagyon jól előrevetítette a szoftverek várható fejlődését, a 3W-t, és itt nem a World Wide WEB-re gondolok.

Azért ne felejtsük el, 1986-ot írtunk, és mindez floppyról ment (tanulhatnának néhány nagy cég programozói).

http://en.wikipedia.org/wiki/GEOS_(8-bit_operating_system

http://www.youtube.com/watch?v=j1Mnvead8Tc

http://www.youtube.com/watch?v=qpX6Tla3U1o&feature=related

A C64-es idők vége felé, erre a gépre a GEOS-on kívül is készült néhány igen komoly, és jól használható alkalmazás, melyek (figyelembe véve a hardver korlátait) igen profi programozói tudásról árulkodtak. A gépből készült hordozható változat is.

Jóval ezután kezdtek terjedni a Public Domain és Shareware programok, amik között nem csak játékokat találhattunk. Ezek egyike sem minősült a szó mai értelmében vett szabad szoftvernek, mert a Public Domain, azaz a közkincs kategória alá eső szoftverek olyan alkotások voltak, amelyeket nem védett szerzői jog, ugyanakkor nem lehetett tulajdonjogot formálni rájuk, és a legtöbb esetben a forráskód sem volt elérhető. Az alkotást bárki szabadon felhasználhatta kereskedelmi vagy nem-kereskedelmi célokra. A GNU filozófiája szerint a Public Domain alá eső szoftver nem valódi szabad szoftver, mert előfordulhat, hogy bár a szoftver igen, a forráskód nem elérhető. Azonkívül a szabad szoftver mindig tartalmaz egy szerzői jogi nyilatkozatot (copyright), ami ez esetben épp a forráskód szabad elérését és másolhatóságát van hivatva biztosítani (copyleft).

A shareware kategóriában általában valamilyen formában korlátozott (idő, funkció) néha erősen bemutató jellegű szoftverek tartoztak, amelyeknek teljes változatát valamilyen (általában pénzbeli) juttatás fejében biztosította a készítő. Forráskódot itt sem tettek elérhetővé a programozók.

A valódi szabad szoftverek megjelenésével és elterjedésével, mindkét kategória, szinte teljesen elveszítette korábbi jelentőségét.

Ideje visszatérni a már említett GNU és GPL témákhoz.

A minket érintő FLOSS rendszerek elterjedése nagyban köszönhető ezeknek a kezdeményezéseknek. Kijelenthető (a korábbi sokfajta kezdeményezést elismerve), hogy csak azok a rendszerek nyújthatnak valódi alternatívát a kereskedelmi szoftverekkel szemben, amelyek világosan kifejezik az alkotók célját a programmal, és hogy mit vállalnak, mire képes a programjuk, kezdeményezésüknek mi a célja.

Ehhez mindenképpen szükség van valamilyen formára, legyen ez a GPL, vagy más hasonló licenc, vagy a GNU kezdeményezésnek való megfelelés.

A következő részben innen szeretném folytatni.

Szőke József

A cikkhez tartozó fórum címe:
http://www.flosszine.org/hogyan_kezdodott

ESR - avagy a bazár ideológusa

"Az egyik stílus – amelyet ma zárt kódú („closed source”) fejlesztésnek hívunk – hagyományos gyári modell, amikor is a vevő egy lepecsételt bithalmot kap, amelyet nem tanulmányozhat, nem módosíthat vagy fejleszthet tovább. A zászlóvivő ebben a megközelítésben a Microsoft. A másik stílus a nyílt forráskód („open source”) – így jött létre az internet is –, amikor is a forráskód elérhető, áttanulmányozható, kölcsönös kódvizsgálatnak vethető alá és gyorsan továbbfejlődhet. A zászlóvivő ebben a megközelítésben a Linux operációs rendszer." - Így nevesítette **Eric Steven Raymond** 9 évvel ezelőtt azt a feszültséget (1., 2.) , amely szerinte az 1960-as évek vége óta ott lappangott a szoftverfejlesztésben. Mióta azonban ESR le is írta a tényt, egyre nyíltabb a két fejlesztőmódszer és ugyanakkor a kétféle szoftverfelhasználói magatartás szembenállása.

Élet, mű

ESR, az Open Source közösség egyik legfőbb ideológusa és programozója, 1957. december 14-én született Bostonban. Saját honlapja (3) és a Wikipedia adatai szerint kb. 30 nyílt projektet vezet, ám előbb megalkotta az INTERCAL programnyelv C-implementációját, gondozta a Fetchmail nevű levelezőprogramot és a Ncurses GNU-könyvtárat, részt vett a GNU Emacs szerkesztő kifejlesztésében, megírta a Linux kernel konfigurálását segítő CML2 nevű eszközt, illetve 1998 februárjában Bruce Perensszel megalapította a Nyílt Forrás Kezdeményezést (4).

1992-ben kezdte írni főműve, "A katedrális és a bazár" (5) egyes fejezeteit. Az addigi szövegeket előbb a Philadelphiai Linux-felhasználók Csoportjának juttatta el, majd 1997. május 22-én a Würzburgban megrendezett, negyedik Linux Kongresszuson nyilvánosan felolvasta. Sokak szerint ez váltotta ki azt a szoftverfejlődés szempontjából forradalmi eseményt, hogy a Netscape 1998. január 22-én bejelentette: nyilvánosságra hozza a Netscape Communicator (a későbbi Mozilla böngésző) forráskódját. Az addig példátlan döntés azt jelezte, hogy egy szoftverfejlesztő cég jobban bízik az önkéntes fejlesztők tízezreiben, mint a saját, fizetésért dolgozó programozóiban. "A katedrális és a bazár" című írás könyvként először 1999-ben jelent meg angolul, fordításai sorra láttak napvilágot. Magyarul a Kiskapu kiadó adta ki 2004-ben, de a mű ingyenes magyar nyelvű kivonata Karsai Róbert tolmácsolásában az interneten is elérhető (6., 7.). Jelenleg a legtöbb nyelven a mű 2000-ben nyilvánosságra hozott, 3.0-ás számú kiadása érhető el. ESR ugyanis következetesen ugyanúgy változtatja meg, egészíti ki a tartalmát, ahogyan szoftvert fejleszt: verzióról verzióra.

Hackermorál

"A Linux felforgató. Ki gondolta volna még csak öt évvel ezelőtt is (1991), hogy a bolygón szétszórt, csupán az internet finom fonalával összekötött sok ezer fejlesztő részidős bütykölése, mintegy varázsütésre, világszínvonalú operációs rendszerré egyesül?" - csodálkozik rá ESR némi álnaivitással "A katedrális és a bazár" elején a kollektív fejlesztés csúcssikerére. És a hangsúly itt a "bütykölés" szón van, amely az angol "hacking" megfelelője és nem a "cracking" szóé, ami "feltörést" jelent. Bár a szóhasználat az angol köznyelvben sem egyértelmű (8.) és a magyar sajtó csaknem minden orgánuma lelkesen azt sulykolja, hogy a hacker számítógépes kalóz volna, azonban a hackerek - Raymond értelmezésében - nem törnek fel és

nem lopnak el semmit, hanem mások által szabadon használható programokat bütykölnek. Nem elvesznek, hanem adnak. ESR sok írása ennek a morális követelménynek - gyakorlati példákkal bőségesen alátámasztott - kibontása, végiggondolása.

ESR "A katedrális és a bazárban" tanulságokat von le a hacker-közösség magatartására és a programfejlesztésre vonatkozóan, több esetben olyan programozókra hivatkozva, mint a Linux-kernel megalkotója, Linus Torvalds (7.), vagy az IBM veterán fejlesztési projektvezetője, Fred Brooks (3.), illetve a popmail klienst fejlesztő Carl Harris (5., 18.). Allegóriájában "katedrális" típusú a zárt kódú, egyetlen cég alkalmazottai által végzett fejlesztés, míg az önkéntes, interneten kommunikáló közösség munkája a "bazár" fejlesztési stílusát testesíti meg.

Tanulságok "A katedrális és a bazár"-ból:

1. Minden jó szoftver egy fejlesztő személyes vágyainak kielégítésével kezdődik.
2. A jó programozók tudják mit írjanak. A nagyok azt is tudják, mit írjanak (és használjanak) újra.
3. „Tervezd be, hogy egyet el fogsz dobni, úgyis el fogod” (Fred Brooks, The Mythical Man-Month, 11. fejezet)
4. Megfelelő attitűd mellett az érdekes problémák megtalálnak.
5. Ha már nem érdekel egy program, az utolsó kötelességed átadni azt egy kompetens utód számára.
6. A gyors kódfejlesztés és a hatékony hibakeresés felé vezető legkönnyebb út a felhasználók társfejlesztőként való kezelésén át vezet.
7. Adj ki korán. Adj ki gyakran. És figyelj a fogyasztóidra.
8. Elegendően sok bétatesztelő és társfejlesztő mellett majdnem minden probléma gyorsan felismerhető, és a javítás is nyilvánvaló valaki számára.
9. A buta kód ügyes adatszerkezetekkel jobban működik, mint a fordítottja.
10. Ha bétatesztelőidet a legértékesebb erőforrásodként kezeled, a legértékesebb erőforrásoddá válva reagálnak.
11. A saját jó ötletek utáni legjobb dolog a felhasználóidtól származó jó ötletek felismerése. Időnként ez utóbbi jobb.
12. A leginkább megdöbbentő és innovatív megoldások gyakran annak a felismeréséből származnak, hogy hibás volt a probléma felfogása.
13. A tökéletességet (a tervezésben) nem akkor érjük el, amikor már nincs mit hozzáadni, hanem amikor már nincs mit elvenni.
14. Akármilyen eszköznek az elvárt módon kell hasznosnak lennie, de az igazán jó eszköz ott is felhasználható, ahol soha nem számítottál volna rá.
15. Bármilyen információközvetítő szoftver írása esetén törekedj arra, hogy a lehető legkisebb mértékben avatkozz csak be az adatáramlásba, és soha ne dobj el semmilyen információt, hacsak a fogadó fél nem kényszerít erre.
16. Ha a nyelved közel sem Turing-teljes, akkor a szintaxissal kifejezőbbé teheted.
17. Egy biztonsági rendszer csak annyira biztonságos, amennyire a titkai azok. Óvakodj az álltitkoktól.
18. Egy érdekes probléma megoldásához találj először egy olyan problémát, amely számodra érdekes.
19. Több ember kétségtelenül jobb egynél, ha a fejlesztés koordinátorának legalább olyan jó kommunikációs közeg áll a rendelkezésére mint az internet, és képes a kényszer nélküli vezetésre.

Az esszé tanulságainak többsége túlmutat a fejlesztéstechnikai alapelvek megfogalmazásán és olyan, munkapszichológiai, valamint gazdasági szempontból a humánerőforrás-gazdálkodók által azóta kiaknázott evidencia, amelyet semmilyen munkaszervezet sem nélkülözhet, ha hatékonyan szeretne működni. Ilyen pl. az 1. és a 18. tanulságban megfogalmazott munka-motiváció kérdése, amely - mint a HR-managerek számára közismert - nemcsak a fizetés nagysága. "A Linux-hackerek által végsőkig fokozott, nem klasszikus gazdasági értelemben vett „hasznos működés” inkább önmaguk kielégítését és a hackertársak közötti megbecsülést célozza." - állítja a szerző. De a 7., a 10., és a 11. tanulság akár a marketing-szakembereket, a 9. vagy a 16. akár a politikusokat, és a kommunikációs igazgatókat is elgondolkodtathatja.

Harcok a gonosz ellen

Raymond más művei (9., 10., 11.) ugyancsak foglalkoznak a zárt és a nyílt szoftverfejlesztés ellentéteivel, és a hacker-közösség kérdéseivel, akár mese formájában mint a Unix Wars (12.): "Valamikor réges-régen egy távoli telepítésen... Kitört a rendszeren belüli háború, mikor is a Felhasználók Szövetsége harcolni kezdett a gonosz Admin Birodalom vasmarkának megtöréséért..." De ESR küzd a gonosz ellen már-már nietzschei magasságokba emelt példabeszédek formájában is, "A Loginataka" c. művében.

A "How To Become A Hacker" vagyis a "Hogyan lesz az emberből Hacker" (13., 14.) c. szövege egyenesen a középkor moralistáira emlékeztet, miközben a programbütykölés normáit ecseteli. Weblapján ő javasolta, hogy a Conway-féle életjáték (15.) siklóreülő (glider) alakzata legyen a hackerek jelképe, ami szintén ESR misztikus vonzódásaira utal.

"Ingyenhardver - trójai faló?" (16., 17.) c. vitairatában ESR nyíltan felveszi a küzdelmet a Sun Microsystems és a Microsoft által meghirdetett számítástechnikai jövőképpel. Ennek előzménye, hogy ő volt, aki 1998-ban nyilvánosságra hozta az ún. Halloween Dokumentumokat (18., 19.), amelyek a Microsoft állítólagos belső anyagai és egyebek mellett a világcég nyílt forráskódú operációs rendszerek elleni taktikáját vázolják fel. Minden harc ellenére a szabad szoftverek ideológusa azonban azt vallja, hogy nem egy cég ellen, hanem egy módszer ellen kell fellépni: a felhasználók fogyasztóvá degradálása ellen.

Anarchia

Mindezen tevékenységeket Raymondnál libertariánus anarchista (20.) meggyőződése alapozza meg, még a megfelelő amerikai pártnak is prominens tagja. "Miért vagyok anarchista" (21., 22.) c. iratában olvasható, hogy "semmilyen, a kormányzati arrogancia kordában tartására tett kísérlet nem válik be - mert az államrend propagandájának hamis ígéretei és mérgező álmai az emberek többségét túlságosan könnyen vezetik a zsarnokság elleni saját, intézményes védelmük feladására." Igaz, ugyanakkor helyesli országa 2003-as iraki inváziójét és egy másik írásából kiderül, hogy a terrorizmus ellen is az állam leépítésével küzdene (23.). Kiáll a fegyvertartás jogáért, mondván, hogy ez az egyén egyik utolsó eszköze a vele szemben szükségképpen zsarnokként fellépő állammal szemben. Sőt, talán a Tae Kwon Do-n alapuló harcművészetnek, a "Moo Do"-nak is ideológiai meggyőződésből vált fekete öves mesterévé.

Most mi van?

Miközben ESR fáradhatatlanul kommentálja a világ eseményeit blogjában (24.) - amelynek címe "Felfegyverezve és veszélyesen", alcíme "Szex, szoftver, politika és lőfegyverek" -, a legutóbbi jelentős hír róla (a cikk írásának időpontjában) lassan már két és fél éves (25.) és arról szól, hogy Fedoráról Ubuntura váltott az előbbi disztribúció függőségi problémái miatt. Két évvel korábban heves kirohanással válaszolt a Microsoft provokációértékű állásajánlatára (26.). Hogy blogján és projektjein kívül könyvírással is foglalkozik-e jelenleg, nem tudni. Mindenesetre a hathatós segítségével létre jött közösségnek jobb lenne, ha nem aprózná el az erőforrásait.

Jankovich Oszkár

Hivatkozások:

1.http://catb.org/~esr/writings/openmind.html
2.http://goliat.eik.bme.hu/~emese/esr/hu/esr.html
3.http://www.catb.org/~esr
4.http://www.opensource.org
5.http://www.catb.org/~esr/writings/cathedral-bazaar
6.http://magyar-irodalom.elte.hu/robert/szovegek/bazar
7.http://www.hwsw.hu/oldal.php3?cikkid=848&oldal=1
8.http://www.merriam-webster.com/dictionary/cracker
9.http://www.catb.org/~esr/writings
10.http://www.catb.org/~esr/faqs
11.http://esr.fsf.hu
12.http://www.catb.org/~esr/writings/unixwars.html
13.http://catb.org/~esr/faqs/hacker-howto.html
14.http://esr.fsf.hu/hacker-howto.html
15.http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
16.http://www.catb.org/~esr/writings/free-hardware.html
17.http://esr.fsf.hu/hwtrojan.html
18.http://www.catb.org/~esr/halloween
19.http://en.wikipedia.org/wiki/Microsoft_Halloween_documents_leak
20.http://www.hayek.hu/ideologiaiszotar2.htm
21.http://www.catb.org/~esr/writings/anarchist.html
22.http://esr.fsf.hu/anarchist.html
23.http://www.catb.org/~esr/writings/against-terrorism.html
24.http://esr.ibiblio.org
25.http://hup.hu/node/35932
26.http://hup.hu/node/9635

A cikkhez tartozó fórum címe:
http://www.flosszine.org/esr_a_bazar_ideologusa

Mondom: U-bun-tu. És csinálom.

Kőbánya, CEU Konferencia Központ. Két év után, az idei első hűvösebb szombaton újra konferenciára gyűlnek az emberszabású lényeknek szánt Linux hazai hívei. A második Ubuntu Konferencián a hallgatóság a disztribúció közösségépítő munkájával éppúgy megismerkedhetett, mint a honosítás, a dokumentálás rejtelmeivel, egyes alkalmazások várható újdonságaival, mások mély rétegeivel, vagy épp a vállalatok Linux-ügyi téveszméivel. Sőt, a szerencsések még (K)Ubuntu telepítő CD-ket, FLOSS könyveket és madárlátta pólókat is zsákmányolhattak a szellemi táplálék mellé.

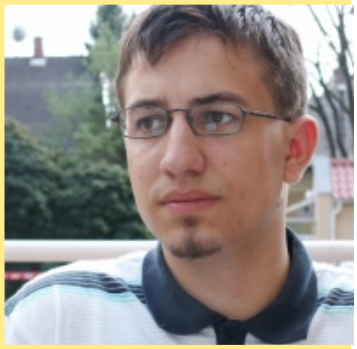
Az üvegajtó szétnyílik és a recepciótól jobbra máris felvehetik a névtábláikat az érkezők. Fehér jár az egyszerű regisztráltaknak, kék a VIP vendégeknek - akik között egyébként a Microsoft egy képviselője is fellelhető. Ott van rendesen a három összekapaszkodó emberből formázott logó mindegyik kitűzőn, ahogyan a szervezők és megannyi Ubi-rajongó pólóján is okker-narancs-vörös körök, s pöttyök ékeskednek. Több ez mint operációs rendszer - ezt sugározza a konferencia egész nap, a 18 előadás legtöbbjén, és a folyosókon is.

"Az Ubuntu egy afrikai gondolat az „emberséggel mások felé” jegyében. Ez a hit abban, hogy van egy univerzális kapocs, ami minden embert összeköt. Ugyanez az elv vezérli az Ubuntu közösség együttműködését. Az Ubuntu közösség tagjainak hatékonyan kell együtt dolgozniuk." - tartalmazza az Ubuntu viselkedési szabályzat, amelyet a Magyar Ubuntu Közösség (1., 2.) is a legjobb tudása szerint igyekszik szem előtt tartani, bár a pénz és a megfelelő számú állandó önkéntes hiánya tavaly nem tette lehetővé a rendezvény megtartását. A 2006 októberi után így a mostani a második magyarországi Ubuntu Konferencia. A szervezők két év alatt a szponzorok számát megnégyszerezték, ami jó esetben megegyezhet a magyarországi Linux-iparban tevékenykedő vállalkozások szaporodási ütemével. A konferenciára idén több, mint 400 résztvevő regisztrált (bejutni csak így lehet), szemben a 2006-os mintegy 350 fővel. A generációs skála a középiskolás korúaktól a csaknem 60 évesekig terjed, a többség 20 és 40 közötti. A szomorú csak az, hogy a gyengébbik nem összesen talán ha tíz fővel képviselteti magát a jeles eseményen.

A párhuzamosan három teremben zajló prezentációk közül több is a hazai Linux-felhasználók, illetve az Ubuntu-közösség együttműködését hivatott erősíteni: Süveg Gábor (3.), a közösségi weboldalak létrehozásának mikéntjét mutatja be, míg Kelemen Gábor (4.), a GNOME kezelőfelület magyarítója kétszer 45 percben, az alapoktól kezdve ismerteti a fordításmenedzsment fortélyait, fordítók, fejlesztők és felhasználók társas viszonyát, amely végül a honosított operációs rendszerben, vagy alkalmazásban képeződik le. Szerinte egyébként az a legfontosabb, hogy mindenki a maga tudásához, nyelvi, számítástechnikai felkészültségéhez mérten vegyen részt a honosító munkában, és tartsa be a lefektetett közös szabályokat, amelyek a megfelelő Wiki oldalakon (5.) és az FSF.hu (6.) dokumentumaiban (7.) elolvashatóak. Kelemen Gábor a csapata és a fordítómunka összehangolására 2005 óta fáradhatatlanul szervezi a fordítóhétvégéket (8.), ahol a munka után, néhány korsó sör hatására, mindig víg pillanatok következnek. A GNOME fordításmenedzsere úgy véli, a nyílt forráskódú programok honosításának egyik legnagyobb akadálya, hogy a világhálón elérhető, szabadon felhasználható angol-magyar terminológiai adatbázisok szűkösek, és képességeik a kommersz szoftverhonosításhoz használt fordítómemóriákétól elmaradnak. Végül kiderül,

Farkas Szilveszter

Negyedéves mérnök-informatikus hallgató a Műegyetemen, az ubuntu.hu közösségi oldal egyik alapítója, a magyar helyi közösség nemzetközi kapcsolattartója, továbbá az EMEA régió Ubuntu Membership Boardjának (**A**) a tagja. A 2006-os Ubuntu Konferencia után idén is főszervezőként vesz részt a rendezvényen. Emellett webfejlesztőként is dolgozik egy kft.-nél, de blogjából (**B**) több minden kiderül.



Hivatkozások:

A - https://launchpad.net/~ubuntu-membership-board-emea

B - http://phanatic.hu

hogy az Ubuntu és a GNOME honosítói a fordításhoz főként KBabelt (9.) használnak, amely a KDE ablakkezelő fordítóprogramja. A GNOME-féle gtranslator (10.) Kelemen Gábor szerint ugyanis "gyönge" a feladatra. Segíti őket még az FSF.hu által életre hívott Mandolin (11.) segédszótár is, amely munkájuk révén állandóan bővül.

Czakó Krisztián a Linux-felhasználók Magyarországi Egyesületének (12.) alapító tagja és elnöke, a Linux Tábor ötletgazdája, a Linux Akadémia (13.) egyik vezetője, a rendezvény egyik támogató cégének is az élén áll. Ő az Ubuntu alatti vékonykliensekről beszél, amelyekkel csak az áramfogyasztást tekintve akár 99 százalékos energiamegtakarítás is elérhető egy vállalati, vagy bármilyen hálózati környezetben, hiszen a munkaállomásokon minimális, a kezelőfelületekhez szükséges konfiguráció fut csupán, míg a rendszer egy központi gépen található, ahol a közös munka is folyik és az adattárolás is történik. Második előadásában Czakó Krisztián beszámol az LME jelenlegi tevékenységéről, végül ötleteket kér a megjelentektől a Linuxok magyarországi népszerűsítésére, majd a legjobbakat pingvines pólókkal díjazza. A hozzászólók legtöbb gondolata a Linuxok iskolákban való elterjesztése körül forog. Egyikük szerint először is egy felmérést kellene elvégezni a középiskolai informatikatanárok körében arról, hogy egyáltalán volt-e már dolguk Linuxszal és hogy oktatják-e, oktatnák-e a felhasználását. Az eredmények birtokában pedig az érdeklődő tanárok iskoláiban bemutatósorozatot, valamilyen roadshowt lehetne szervezni, ötletel tovább valaki a közönségből. A legkézzelfoghatóbb felvetés talán az, hogy az egyetemek és a főiskolák sokasodó diákújságait is érdemes volna linuxos írásokkal megkeresni, ugyanis a legtöbb szívesen közzé tenne ilyeneket. Az LME egyébként a wiki.lme.hu-ra és a wiki.linux.hu-ra is várja a további ötleteket.

A konferencia gazdája, az FSF.hu nemcsak a Magyar Ubuntu Közösséget segíti, hanem több hazai nyílt forráskódú projektet, így a FLOSSzine honlapjának is a saját, "barack" névre hallgató szerverrendszerén ad helyet. Mátó Péter (14.), az alapítvány vezetőségi tagja a főszponzor munkatársa, ennek a virtuális szolgáltatásfarmnak a történetét, a felépítését és az üzemeltetését taglalja, különös tekintettel a virtuális kiszolgálókhoz szükséges erőforrások megtervezésére, takarékos kihasználására és biztonsági követelményeire. Miközben olyan kulcsszavak röpködnek, mint KVM, VMware, VirtualBox, Quemu és KQuemu, vagy Xen, példásan sok, szemet nyitogató apróság is eljut az érdeklődőkhöz, olyan hasznos dolgok, mint hogy a virtuális gép architektúrájának nem kell megegyeznie a gazdagép valós architektúrájával.

A déli szünetben azután egy kissé lassan fogy a sor a kiváló és olcsó menzán, de szerencsére marad idő az Örs vezér térig is eljutni, ahol a legközelebbi alternatív étkezdék találhatóak. Azonban van olyan ifjú diák, aki ekkor inkább ráveti magát az aulában tanyázó, Hardy Heronnal (15.) felszerelt gépre és azonnal érdeklődik is az egyik támogató céget képviselő nyakkendős úrtól: "Milyen programok vannak Ubuntura?" Mivel a válasz - "Mindenféle" - nem elégíti ki, a fiú tovább kérdez, hogy "És hány program van Ubuntura?". Ekkor az úr kinyitja a Synapticot és amikor 24 ezer 668 csomag van kilistázva, gyorsan biztosítja róla a meghökkent srácot, hogy "azért ez nem mind önálló felhasználói program, sok segédprogram is van közötte, tehát olyan tízezer körül".

A nap második felében először is Molnár Vilmos avat be kissé szűkszavúan az Ubuntu alatti mobilinternet-használat miértjébe. Mint mondja, szerinte Magyarországon általában a VMC (16.) használata a legjobb megoldás, bár telefonszolgáltatója válogatja. Megnyugtat mindenkit, hogy a nemsokára megjelenő,

következő Ubuntu disztribúció, az Intrepid Ibex már jobban fogja támogatni a mobil internetkapcsolatokat. Az elmondottakhoz kapcsolódva egy laptop és egy mobilstick segítségével létrehozott kapcsolódás viszontagságairól mesél vidáman az egyik hallgató: az M7-es autópályán buszozva remekül működött az internetezés, mindaddig, amíg a jármű el nem érte a bűvös óránkénti 80 kilométeres utazósebességet. Akkor ugyanis az internetes kapcsolat mindig megszakadt, majd mielőtt újra felépült volna, újra csak megszakadt és ezután meddővé vált a próbálkozás. Akkor a fiatalember megkérte a sofőrt, ugyan lassítana-e le ismét 80 alá, és láss csodát: a kapcsolat újra folyamatosan működött. Következtetés: a mobilinternet-szolgáltató bázisállomásai olyan sűrűn vannak telepítve az autópálya mentén, hogy egy mai, Ubuntuval felszerelt átlag-laptop nem tud elég gyorsan csatlakozni az egymást követő jelforrásokhoz a 80-as sebességhatár fölött. Vagy célja a szolgáltatónak a balesetmegelőzés is, vagy a jövő laptopgenerációjába fektetett be az adótornyok besűrítésével - hangzottak a kuncogó értékelések.

Ezek után a nagyteremben kisebb meglepetés. Kruska Tibor IT menedzser úgy tart előadást a vállalati Linux-megoldásokról és az SAP-ról (17.), hogy nem jelenik meg a teremben. Azaz, a prezentáció végén mégis megjelenik - a kivetítővásznon. A kötelesség ugyanis egy valamivel nagyobb eseményre szólította, épp Amerikában vesz részt egy SAP konferencián. Előadását azért előzőleg eljuttatta a Magyar Ubuntu Közösségnek, szóval Enter. Mint az egyik jelentős szponzorcég vezetője, ő sem marad adós az ínyenc történetekkel, miközben azokat a Linux bevezetése melletti érveket sorolja, amelyekkel vállalati ügyfeleinél a legtöbbször célba talál. Kiderül, hogy jelenleg is van olyan magyar online médiacég, amely már Linux+SAP konfiguráció mellett döntött, de az is, hogy a Magyar Államvasutak belső telefonrendszerében (ezek azok a vonalak, amelyeken a baktert nemrég még felcsöngették a legjobb álmából, hogy idejében engedje le a sorompót) ma már Asterisk (18.) segítségével jönnek-mennek a vasúti hírek. Megtudjuk még, hogy a Nagios (19.) már egy olyan globális cégnél is hadrendbe állt központi monitorozó rendszerként, mint a Nike.

Akár világméretű hálózatokon futó, komplex rendszerek kiterjedt és szabályozott naplózására is jó a syslog-ng (20., 21.) - állítja a színpadon az alkalmazás egyik hazai szakértője, Höltzl Péter, és ezt azzal indokolja, hogy a program formátum, protokoll, felépítés, forrás- és célmeghajtók, logszervezés, flagek, szűrők, valamint makrók szempontjából is megfelelően alakítható ahhoz, hogy a különböző szervezet-specifikus igényeket kielégítse. Aztán a statisztikai spamszűrés elméletéről és gyakorlatáról, vagyis a tanítható, és maguktól is tanuló spamszűrő programok működéséről és biztonságáról a FLOSSzine egyik szerzője, Sütő János ad elő. Számtalan kéretlen e-mail fejlécét elemezve egyértelmű, hogy a levélosztályozás elvén megvalósuló szűrés egy körülbelül egy hónapos, átlagos napi használat mellett, 99 százalékosnál is nagyobb biztonsággal vethető be bármilyen trükkös spammerek támadásai ellen.

A konferencia szervezői végül környezettudatos meggyőződésükről is tanúbizonyságot tesznek. Felkérik a jelenlevőket, hogy a névtábláikat ne vigyék el magukkal emlékbe, hanem szolgáltassák csak szépen vissza. Ezért a kisorsolt szerencséseknek ajándék is jár, Torma László, Az Ubuntu világa című könyvének (22.) immár nyomtatásban is megjelent változata. A szervezők ugyanis remélik, hogy ugyanezek a kitűzők a jövő évi Ubuntu Konferencián is használhatóak lesznek.

Jankovich Oszkár

A cikkhez tartozó fórum címe:
http://www.flosszine.org/ubuntu_konf2008

Hivatkozások:

1 http://ubuntu.hu/worldoflinux/part10
2 http://ubuntu.hu/konyvlap/kozosseg#dok
3 http://gabor.suveg.eu
4 http://kelemeng.blogspot.com
5 http://wiki.hup.hu/index.php/Fordítás_HOGYAN/Bevezetés
6 http://www.fsf.hu
7 http://forditas.fsf.hu/html/Utmutato.html
8 http://ubuntu.hu/hirek/2008sep/forditohetvege-809
9 http://kbabel.kde.org
10 http://gtranslator.sourceforge.net
11 http://mandolin.midian.hu/glossary
12 http://www.lme.hu
13 http://www.linuxakademia.hu/udvozoljuk
14 http://www.fixme.hu
15 http://hu.wikipedia.org/wiki/Ubuntu_(Linux-disztribúció)
16 http://ubuntu.hu/blog/hotplug/vmc-10
17 http://hu.wikipedia.org/wiki/SAP_AG
18 http://www.asterisk.org
19 http://www.nagios.org
20 http://en.wikipedia.org/wiki/Syslog-ng
21 http://www.balabit.hu/network-security/syslog-ng
22 http://toros.hu/ubuntu/az-ubuntu-vilaga.pdf

Ubuntu@Hu

Az Ubuntu még mindig igen fiatal Linux disztribúciónak számít: az első kiadás nem egészen négy éve, 2004. október 20-án jelent meg. Mára azonban az egyik legnépszerűbb disztribúcióvá vált, rengeteg felhasználóval és lelkes közösséggel. Magyarországon pedig különösen népszerű, egyre több oldalon láthatjuk a bannereit, rendszeresen olvashatunk róla az on-line sajtóban, az ubuntu.hu oldalnak pedig több mint hatezer regisztrált felhasználója van.

Mi a népszerűség titka? Hogyan tudott ez a fiatal disztribúció ilyen hamar ennyire sikeressé válni? Talán valami afrikai vudu mágia van a háttérben? Nos, bár mágiáról szó sincs, azonban az afrikaival nem is járunk annyira messze az igazságtól. Az Ubuntu ugyanis egy ősi afrikai kifejezés, melynek jelentése: "emberséggel mások felé" (ellentétben azzal a rosszindulatú és teljességgel alaptalan feltételezéssel, amely szerint azt jelentené, hogy "nem tudok Debiant telepíteni").

Persze sokan most nyilván értetlenkednek, hogy mi köze lenne egy GNU/Linux disztribúciónak ehhez a szép, de igencsak elvont gondolathoz. Nos az, hogy az Ubuntunál ezt az elvet tudatosan beépítették a népszerűsítésbe. Felismerték ugyanis, hogy a közösségi alapú modellek nem csak a szoftverek fejlesztésében működnek remekül, hanem a népszerűsítésében is. Ezzel pedig visszanyúltak a gyökerekhez, hiszen a Linux a kezdet kezdetén, mikor nem álltak még sikeres vállalkozások mögötte, ugyanúgy kizárólag a lelkes felhasználóknak köszönhetően terjedt.

Az Ubuntu mögött egy nyereség orientált vállalkozás, a Canonical áll, amelynek nyilvánvalóan az a célja, hogy hatékonyan bevételt termeljen. Ehhez pedig elengedhetetlen a népszerűség. Minél többen használnak Ubuntut, annál többen vásárolnak hozzá terméktámogatást, annál több szakembert kell képezni, annál több cég fizet jutalékot, hogy hivatalos Ubuntu partner lehessen, és annál többen kortyolják reggeli kakaójukat Ubuntus bögréből. Az Ubuntu felismerte azt, hogy drága reklámok, nyakkendős, túlfizetett salesesek hadserege és folyamatosan meetingelő marketingesek nélkül is el lehet érni a felhasználókat - mégpedig úgy, ha közvetlenül hozzájuk fordulnak. Az Ubuntu óriási hangsúlyt fektet az úgynevezett helyi felhasználói közösségekre, vagyis LoCo-kra. Ma már szinte minden nagyobb országban működnek ilyen társaságok, melyek kiemelt figyelmet, és hasznos erőforrásokat kapnak: a hivatalosan elismert közösségeknek saját levelezőlistája, IRC csatornája van, és minden új Ubuntu kiadás megjelenésekor ingyen kapnak egy nagy dobozt, több száz telepítő CD-vel.

Az Ubuntu sikerének a kulcsát ezek a közösségek jelentik. Ezen közösség tagjai végzik az Ubuntu lokalizációját, tartják fent a weboldalakat, írják a dokumentációt, népszerűsítik baráti körükben az Ubuntut, használják a munkahelyükön, cégüknél, teszik ki a bannereket a weboldalukon, töltenek fel látványos videókat a YouTube-ra, vagy blogolnak kedvenc disztribúciójukról. Az Ubuntu pedig megjelenésétől arra törekedett, hogy odafigyeljen ezekre a helyi közösségekre, segítse a létrejöttüket. A félévente küldött jókora doboz, tele CD-kel, kézzelfoghatóvá teszi ezt a kapcsolatot.

A hazai közösségnek nemzetközi viszonylatban sincs oka szégyenkezni, több tucat ember vesz részt a munkában kisebb-nagyobb rendszerességgel, az Ubuntu.hu oldalnak több mint hatezer regisztrált felhasználója van. Mára az Ubuntu LoCo az egyik legnagyobb és legaktívabb hazai szabad szoftveres közösséggé nőtte ki magát. A magyar Ubuntu közösség végzi a disztribúció honosítását, saját dokumentációt tart karban, amit kiadásról kiadásra aktualizál, rendszeresen

frissített weboldalt, IRC csatornát, közösségi és segítségnyújtó levelezőlistát üzemeltet, konferenciát, sörözéseket szervez, sajtóközleményeket ír, bannereket készít, és fesztiválokon népszerűsít. Vagyis a magyar Ubuntu közösség éppen úgy működik, mint egy igazi szabad szoftveres projekt, azonban itt már elsősorban nem a programozói kvalitásait használja az ember, hiszen az Ubuntu fejlesztését nem mi végezzük. A Linux fejlesztésében már bevált, decentralizált, aktivisták lelkesedésére aktívan építő bazár-modell azonban ugyanolyan jól működik a honosításban, dokumentációk készítésében vagy éppen a népszerűsítésben. A útmutatók karbantartásánál például pontosan ugyanazt a Bazaar verziókezelő rendszert használjuk, mint amivel sok szabad szoftver fejlesztését is koordinálják.

Az Ubuntu közösségben végzett munkájáért senki sem kap pénzt - de ezt nem is ezért csináljuk. Az alkotás öröme, a közös munka élménye és persze a hírnév motiválja az embereket. No meg a szabadság: az Ubuntu közösségben ugyanis mindenki önmagát valósíthatja meg. Ez nem egy munkahely, ahol az embereket olyasmire kényszeríthetik, amihez semmi kedve (legyen az uránbányászat vagy egy egyszerűsített éves beszámoló elkészítése). Ugyanakkor az emberben mégis kialakul a kötelességtudat: bár nem a legmókásabb feladat például a dokumentáció aktualizálása egy új kiadásra, vagy a képernyőképek tömegének lecserélgetése - például a csúszdázás, vagy a konfetti szórás sokkal vidámabb dolog, de mégse merül fel bennünk, hogy ne csináljuk meg. Ugyanis arra nagyon büszkék vagyunk, hogy nálunk már egy új kiadás megjelenésének napján az összes fontos útmutató elérhető hozzá, ez pedig csak megér annyit, hogy pár estét rászánjunk az időnkből.

Ahogy pedig nő a közösség mérete, egyre nagyobb a felelősség is. Ma ha valaki Magyarországon életében először próbálja ki a Linuxot, akkor az jó eséllyel egy Ubuntuval fog próbálkozni. Éppen ezért rajtunk is múlik, hogy később hogyan fog viszonyulni a Linuxhoz. Ha az Ubuntu közösséggel jó tapasztalatokra tesz szert, a problémáira könnyen megtalálja a megoldást a dokumentációban, a kérdéseire pedig segítőkészen reagálnak a fórumban, akkor sokkal kisebb az esélye, hogy csalódva távozik. Elképzelhető, hogy 1-2 év múlva majd vált egy bütykölősebb, komolyabb Linuxos ismereteket igénylő disztribúcióra - például Archlinuxra vagy Gentoora. Ehhez azonban az kell, hogy az elején ne szegjük kedvét.

Az Ubuntu egyre szélesebb körben terjed. Míg korábban a Linuxot a rendszergazdák, informatikus hallgatók és kísérletező kedvű, lelkes amatőrök operációs rendszerének tartották, mára a felhasználói bázis ennél jóval sokszínűbb. Bár a nagy áttörés még nem történt meg, hiszen még mindig jóval többen használnak Windowst, mint Ubuntut, azonban mi már nem engedhetjük meg magunknak azt a luxust, hogy úgy kezeljük a felhasználókat, mintha ők is profik lennének. Az én ismeretségi körömben van olyan 11 éves kislány, aki nagyon ügyesen használ a saját gépén Ubuntut - nincs is vele különösebb problémája, de ha segítséget kérne a fórumban, akkor egyszerűen nem adhatjuk neki azt a választ, hogy "RTFM".

Az Ubuntu közösségnek fel kell arra készülnie, hogy a segítséget kérő már nem feltétlen egy húsz éves, BME infó szakos hallgató. Lehet, hogy egy általános iskolás kislány, vagy egy nyugdíjas bácsi nem boldogul valamivel. Azt pedig, hogy ennek a kihívásnak hogyan tud egy decentralizált, a szabad szoftverek fejlesztési modellje mentén szerveződő közösség megfelelni, még mi is csak tanuljuk. Nekem semmi kétségem afelől, hogy ez sikerülni fog - de azzal is pontosan tisztában vagyok, hogy ezért még tennünk kell. Már csak azért is, hogy felnőjünk az Ubuntu szellemiségéhez: "Emberséggel mások felé".

Torma László

A cikkhez tartozó fórum címe:
http://www.flosszine.org/ubuntu_at_hu

Warsow

Manapság az FPS játékstílus trendinek is mondható: kisebb-nagyobb csapatok tömkelege fejleszt szabad és kereskedelmi jellegű megvalósításokat. A célközönség válaszként mérhetetlen játékidővel rója le hálóját - persze csak akkor, ha az elkészült mű eredeti, ötletes és összességében is élvezetes.

A fő erények

Három fő erényt említettem a bevezetőben - azt a triót, ami sajnos nem túl gyakran lelhető fel egy „ismeretlen” programban. Sőt, ha az élvezetes jelzőt tovább boncolgatom, akkor még ennél is jobban leszűkül a kör. Hiszen mitől lehet élvezetes egy játékszoftver? A választ szerintem jól tudja minden Olvasó: szubjektív megítélés alapján a látványvilág, a fizika, az összetettség és a kihívás kombinációja bárkinek elnyerheti a tetszését. Nos, a címben szereplő projekt valószínűleg minden vérbeli FPS rajongó ízlésvilágában mély nyomot fog hagyni: ritka módon eredeti próbálkozás - ráadásul ingyenesen elérhető és Linuxon is fut. További erénye, az alapjául használt Qfusion kód (a Quake2 motorjának egy modifikációja), ami által garantált a pörgős játékmenet, miközben a megvalósítás kiforrott, gyökerei tíz évre nyúlnak vissza. Úgy gondolom, ember ennél többet nem kívánhat...

Bővebben

Látogassunk el a Warsow hivatalos honlapjára, a http://www.warsow.net címre! A rövid ismertetőből kiderül, hogy egy erősen sport-FPS jellegű, többjátékos üzemmódra kihegyezett lövöldözős játékról van szó. Sajnos a bevezetőben nem hangsúlyozzák eléggé, de a program szerencsére rendelkezik beépített botrendszerrel, a programozók gondoltak a hálózatok és az internet lehetőségeit nélkülöző felhasználókra is. Ezáltal bárki beállíthat gépi ellenfeleket önmaga vagy csapata ellen, esetleg a „foghíjas” gárdák hiányzó tagjait is pótolhatja velük (természetesen nem árt észben tartani: AI ide vagy oda, a mesterséges játékosok sohasem versenyezhetnek a hús-vér ember irányította karakterekkel). Ha valaki veszi a fáradságot és megnéz még néhány képernyőmentést is, akkor talán minden kétsége elszáll az egyediséget illetően: a képregényszerű, neon hatású grafika szinte vonzza a tekintetet (a megjelenés vészesen hasonlít egy grafikus leképezési eljárás, a Cellshading algoritmus szakszerű használatának eredményéhez). A fotók terén nem csak a küllem érdemel szót. Az ott látható játékosok sokszor „pózolnak” lehetetlen helyzetben, így nyomatékosítva, hogy a műfajra jellemző trükkök (pl. Rocket Jump, Wall Jump, Ramp Slide) mesterei itt aztán kiélhetik minden hajlamukat. A reflexeket DeathMatch, Team DM, Capture The Flag, Duel, Race és Midair módokban lehet kamatoztatni, ezek az ismert stílusok valószínűleg nem hoznak majd senkit sem zavarba. Ha a weblapon írtakon túl próbálnám meg bővebben áttekinteni a Warsow sajátosságait, akkor menthetetlenül elfogult lennék: „Itt kérem szépen, minden a helyén van”.

Mindezt hogyan?

A programkód linuxos verziója az előbb írt, hivatalos URL mögött érhető el. Üzembe állítása két módon történhet: az első

1.ábra A játék, KDE asztalon

2.ábra Remek pályák, egyedi látvánnyal

megoldás szerint a warsow_verzió_zip állományt töltsük le valamelyik tükörszerverről, majd csomagoljuk ki egy tetszőleges, írható területre. A játék főkönyvtárában több rendszerhez kötődő bináris található (Linux i386, Linux x64, Win32, Win64, Portable - valamint az ezekhez kapcsolódó szerver üzemmódok állományai), melyek közül a kívánt indító szkript (warsow, wsw_server) kiválasztja a nekünk leginkább megfelelőt. A második út a jól ismert Loki Installers for Linux Gamers csapatához vezet, akik Loki alapú telepítőbe „drótozott" kiadást készítettek az archívból. Látogassunk el a LIFLG oldalára, a http://www.liflg.org címre ! A natív szekcióból kitallózható warsow_verzió_nyelv.run állományt töltsük le, majd root jogkörrel indítsuk el! A szkript grafikus felületen és konzolon egyaránt használható: csupán annyi dolgunk akad, hogy a licencet elolvassuk, valamint az alapértelmezett elérési utat (/usr/local/games/warsow) jóváhagyjuk. Ha minden fájl a helyére került, a telepítő létrehoz egy linket az /usr/local/bin mappában warsow néven, így a játék felhasználóként kiadott warsow parancsra indul. Ezután minden személyes beállítás kulturált módon, a ~/.warsow rejtett könyvtárban lesz fellelhető. Természetesen tisztán szerver üzemmódban is indítható mindez, ./wsw_server parancsot kiadva a program mappájában. Bármelyik megoldást választjuk, a projekt helyigénye mindössze 250Mbyte lesz, a kompromisszumoktól mentes használatához pedig egy 1500MHz órajelű (teljes értékű) x86 CPU, 256MByte memória, és egy GPU szerepet betöltő 3D grafikus kártya szükségeltetik (működő GLX, vagy DRI kapoccsal). Ezeket a szerény paramétereket akár még a három évvel ezelőtt megvásárolt PC-k is teljesítik, ebből adódóan a Warsow a „kiöregedőben lévő", otthoni számítógépeken is szépen fut. A *.run állományról mindenképpen meg kell még jegyeznem, hogy nem feltétlenül naprakész kivitelezésű, valamint a hangszolgáltatásért felelős meghajtók csak az SDL_AUDIODRIVER környezeti változó megfelelő értékre(alsa, artsc, dma, esd, nas) való állításával cserélhetőek, amit az export parancs segítségével tehetünk meg. Az utóbbi információra sokaknak szüksége lehet!

Tapasztalatok

Már kezdetben is magas elvárásokat támasztottam a Warsow-val szemben. Aztán eltelt néhány hónap és még mindig nem tudtam félretenni ezt a projektet: szólóban, csapatban, klánokhoz csatlakozva is megunhatatlan. A grafikai kivitelezés valóban unikum. Nem mellékesen villámgyors. A játékmenet pörgős, tisztán érezni a Quake2 bizonyított megoldásait, ráadásul a műfajra jellemző mozgástechnikai trükkök fizikája pillanatok alatt kiismerhető. Az egészben talán az a leginkább mellbevágó, hogy a cikk írásakor fellelhető aktuális verzió még csak v0.42 jelölésű, tehát a szoftverben még nagy potenciálok rejlenek, érdemes lesz komolyan odafigyelni rá. Már most, ilyen fejlettségi szint mellett is klánok alakultak a hatékony csapatmunkára - ezek között természetesen fellelhetőek olyan társaságok is, melyek Linux-felhasználókból verbuválódtak. Megéri meglesni a játékot (szakértő kezek általi) mozgásban is, erre a http://youtube.com „médiagyűjtő" tökéletesen megfelel: a warsow keresési kulcsra számtalan érdekes találatot kaphat az érdeklődő.

Kovács Zsolt

A cikkhez tartozó fórum címe:
http://www.flosszine.org/warsow

3.ábra Közelharc készülődik

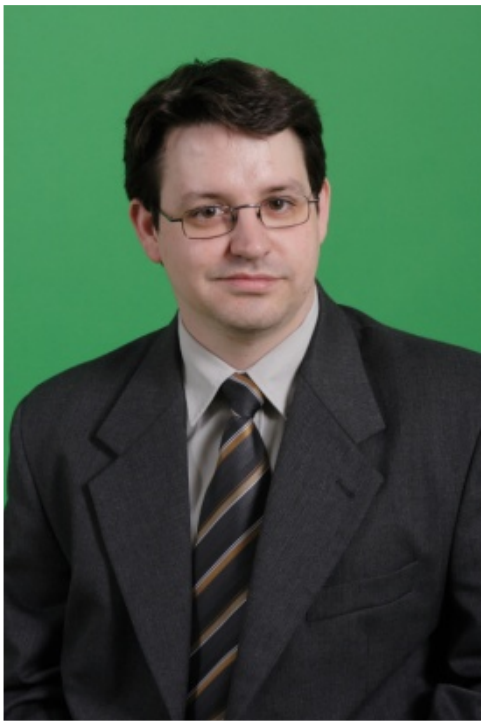
4.ábra Apró győzelem

Scheidler Balázs (syslog-ng)

-FLOSSzine: *Miként jött a gondolat, hogy ebbe belefogj?*

-Scheidler Balázs: Fontos tudni, hogy a syslog-ng első változata 1998-ban készült, amikor 3. éves egyetemista voltam. Egyetemistának pedig sok ideje van és kevés pénze. A sok időmet a Linuxszal való ismerkedéssel töltöttem - naponta fordítottam kernelt például. A pénz problémát pedig kisebb-nagyobb projektek segítségével próbáltam orvosolni.

Az ismerkedés kapcsán találkoztam a syslogd-vel, és mindig problémám volt azzal, ahogy a syslogd a különböző naplófájlokba az üzeneteket szétszedte: sosem találtam semmit. Általában az első dolgom volt Linux telepítés után, hogy az összes rendszerüzenetet ugyanabba a fájlba irányítottam.



Ekkor jött az MLF-es Linux listára egy felhívás, hogy az nsyslog nevű programot kellene testreszabni. Ez lett a syslog-ng 1.0.

Röviddel később aztán úgy döntöttem, hogy újraírom az nsyslogból örökölt kódot, így lett a syslog-ng 1.2, ami már az én agyszüleményem volt.

-FLOSSzine: *Miért döntöttél a nyílt forrású megvalósítás mellett?*

-Scheidler Balázs: Akkoriban kerestem, hogy milyen szabad forráskódú projektet indíthatnék. Több próbálkozásom is volt, de szerencsémre a syslog-ng kapcsán a megrendelőm kikötötte, hogy a végeredménynek nyílt forrásúnak kell lennie. Tulajdonképpen e szempontból szerencsém volt a megrendelővel.

-FLOSSzine: *Ennek a döntésnek azóta mik a hozományai?*

-Scheidler Balázs: A syslog-ng rengeteg elismerést és ismertséget hozott. Nagyon jó érzés, ha az ember produktumát sokan használják.

-FLOSSzine: *Miként befolyásolta ez a termék fejlődését?*

-Scheidler Balázs: Mivel a szoftver ingyenes, ezért rengeteg visszajelzést kaptam a felhasználóktól. Engem pedig az motivált, hogy újabb és újabb felhasználókat nyerjek meg a syslog-ng-nek. Így ezeket a visszajelzéseket általában gyorsan egy-egy patch követte. A felhasználóim hálásak voltak, én meg ennek örültem.

-FLOSSzine: *Miként befolyásolta ez a termék eladhatóságát?*

-Scheidler Balázs: A syslog-ng közel egy évtizeddel előzte meg a korát, az első kereskedelmi szoftverek ezen a területen 2-3 éve jelentek meg, és majdnem mindegyik prospektusában benne van, hogy „syslog-ng kompatibilis”.

A syslog-ng nemrégiben megjelent kereskedelmi ága, a Premium Edition értékesítése globális. Kis magyar cégként adtunk el syslog-ng-t minden kontinensen, személyes jelenlét nélkül.

Úgyhogy úgy gondolom, hogy kereskedelmileg is megérte.

-FLOSSzine: *Melyik a kedvencebb „gyerek”, a syslog-ng vagy a Zorp?*

-Scheidler Balázs: Hmm... jókat kérdezel. A Zorp egy nagyságrenddel nagyobb, és sokkal komplexebb. Mindkettőt szeretem, főleg az a jó, hogy időnként válthatok a kettő között.

-FLOSSzine: *Hogyan viszonyul a syslog-ng PE (Premium Edition) az OSE (Open Source Edition) változathoz? Milyen fejlesztési irányvonal jellemzi őket?*

-Scheidler Balázs: A syslog-ng Premium Edition keretében azokat a funkciókat kezdtük el megvalósítani, amit a nyílt forráskódú verzióban már többen kértek, viszont implementációjuk nagyobb falat volt, mint amit én hobbiként be tudtam volna vállalni. A syslog-ng 2.1-es verziójánál a Premium és az Open Source Edition között kicsi volt a különbség.

Most készül a syslog-ng PE 3.0-ás verziója, itt már a BalaBit hátterét is jobban kihasználva, több fejlesztő, és tesztelő dolgozott. Az újonnan fejlesztett funkciók egy részét megjelentetjük az Open Source-ban is.

-FLOSSzine: *Milyen újdonságok várhatóak a syslog-ng új verziójában (3.0)?*

-Scheidler Balázs: A legfontosabbak talán: az IETF specifikáció alatt álló új protokollok támogatása az üzenet tartalmának feldolgozása, a syslog-ng képes lesz információt kivenni az üzenetből és felhasználni azt az üzenet tárolásánál, továbbításánál üzenetek tartalmának módosítása a 2.1-es kereskedelmiből publikálásra kerül az SQL adatbázisba való naplózás.

A kereskedelmi változat ezen felül további funkciókkal bővül, talán a legfontosabb a titkosított, tömörített naplófájl formátum.

A változások jelentős része inkább a részletekben rejlik, az új módosítások nem elszigetelt funkciók, a teljes egész egy olyan rendszerként dolgozik együtt, amivel komplex problémákat is egyszerűen lehet megoldani.

-FLOSSzine: *Ma hogyan indítanád el ugyanezt a projektet, változtatnál-e bármin?*

-Scheidler Balázs: Kommunikáció és marketing. Egy FLOSS projektnek is szüksége van ilyesmire, én azonban ezeket mindig másodlagosnak tekintettem, és a mai napig nincs rá időm. A BalaBit marketingjét pedig nehéz erre fókuszálni, ők egy sokkal inkább kereskedelmi modellben gondolkoznak.

A syslog-ng community nagyon jó, de ez elsősorban a levelezési listán jelenik meg, a weben nem igazán.

-FLOSSzine: *Mennyiben mozdítja elő egy jó nevű termék/jól prosperáló cég a FLOSS ügyét?*

-Scheidler Balázs: Egy sikeres projekt további fejlesztőket és felhasználókat hoz be, ilyen módon erősíti a közösséget.

-FLOSSzine: *Mik a jövőbeli tervek? Appliance termékkészítés szerepel-e a listán?*

-Scheidler Balázs: Igen, a syslog-ng alapú appliance-t a PE 3.0 -s fejlesztésével párhuzamosan készítjük, és nagyjából egyszerre fognak megjelenni.

A cikkhez tartozó fórum címe:

http://www.flosszine.org/interju_scheidler_balazs_syslog-ng

syslog-ng

Hello Window!
GTK+/gtkmm programozás GNU/Linux alatt

A GTK+ (GIMP Toolkit) egy C nyelven -ám objektum-orientált megközelítéssel- íródott, grafikus felhasználói felületek (GUI) létrehozására használatos alkalmazás-programozási interfész. A gtkmm nem más, mint ennek a függvénykönyvtának a C++ változata, pontosabban foglamazva wrappere. Mindkét terjesztése LGPL licensz alatt történik, így bátran felhasználható mind szabad/ingyenes, mind kereskedelmi szoftevrek létrehozására.
A most kezdődő sorozat célja az abszolút kezdetektől indulva bemutatni a GTK+ és a gtkmm hasonlóságait, különbözőségeit, sajátosságait eljutva egy olyan szintre, ahol remélhetőleg a több éves tapasztalattal rendelkező fejlesztők is találnak hasznos, megfontolásra érdemes ötleteket, információkat.

1 Bevezetés

A cikksorozat ezen első részében a GTK+/gtkmm programozás szempontjából teljesen az alapokról indulunk, de feltételezünk némi jártassságot a C, illetve C++ nyelvű programozás, illetve a Linux alatti fejlesztés terén. Utóbbiak tekintetében javasolt *Vomberg István*, *Programozás Linux környezetben* című cikksorozatának tanulmányozása, hisz számos területet érintünk az ott tárgyaltak közül anélkül, hogy itt részletekbe menő kifejtésük megtörténne.

2 Összehasonlítás

Mielőtt a kódolás technikai részleteire rátérnénk, tegyünk egy rövid kitérőt a tisztánlátás érdekében és hasonlítsuk össze a két felületprogramozási nyelv alapvető tulajdonságait:

	GTK+	**gtkmm**
Felhasználási terület:	GUI fejlesztés	GUI fejlesztés
Implementáció nyelve:	C	C++
Implementáció módja:	natív	wrapper
Objektumorientált technikák használata:	közvetett	natív
Típusellenörzés:	futási időben	fordítási időben
Licenc:	LGPL	LGPL
Ismertebb projektek:	GNOME, Evoluion, Firefox, Gimp	GNOME

2.1 GTK+ vs. gtkmm

Ahogy az a fentiekből is látszik a a C++ nyelvű változatnak megvannak a maga komoly előnyei. Ezek közül talán a legfontosabb, hogy a nyelv nyújtotta módszereket, mint például az öröklöltetés, itt közvetlenül használhatjuk ki. Ez természetesen nem jelenti azt, hogy a GTK+ esetén erre ne lenne lehetőségünk, ugyanakkor meg kell jegyezni, hogy míg a gtkmm esetén ez játszi könnyedséggel megtehető, addig az eredeti változat alkalmazásával ez kissé körülményes. Egy másik megfontolandó érv a típusbiztosság, melynek előnyeit nem lehet eléggé hang-

súlyozni, hiszen hosszú órák hibakeresésétől óvhat meg minket. A GTK+ a GObject révén -mellyel egy későbbi részben részletesebben is foglalkozunk majd- rendelkezik egy frappáns mechanizmussal, mely lehetővé teszi a futásidejű típusellenőrzést a widgetek esetén, de nem vetekedhet a C++ nyújtotta fordítási idejű hibaüzenetekkel.
Megfeledkezni azonban nem lehet a tényről, hogy a C++ fordító közel sem olyan gyakran áll rendelkezésre, mint azt gondolnánk. Számos olyan terület létezik ugyanis, melyek esetén alapvető megkötés a C nyelv. Megemlítendő továbbá, hogy a gtkmm nem az egyetlen port, hiszen létezik többek között python (pyGTK), perl (gtk2-perl), ruby, java nyelvű változat is.

2.2 Kódszervezés

Elöljáróban fontos lehet még tudni a nyelvet megvalósító implementáció szervezéséről, hogy

példaértékűen választja szét a funkcinalitás egyes elemeit több különálló, jól elhatárolt részegységre, melyek ugyan támaszkodnak egymásra, de a megoldás nagyban elősegíti a rugalmasságot és a portabilitást. A GDK (GIMP Drawing Kit) felelős a rajzolási primitívek megvalóításáért, tulajdonképpen nem egyéb, mint egy wrapper az ablakozó rendszer köré. Hasznossága abban áll, hogy amennyiben sikerül portolni azt egy új grafikus környezetre, akkor a GTK+ máris működőképessé válhat. A GObject, mely a GLib része minden widget "ősosztálya", ugyanakkor saját eszközök alapkövévé is thetjük, hisz egyebek mellett lehetővé tesz referenciaszámlálást, futásidejű típusellenőrzést, tulajdonságok hozzárendelését és azok értékének futásidőben történő változtatását.

2.3 GLib

A GLib maga egy önálló függvénykönyvtár, mely számos hasznos -a C nyelv elemeként nem létező- programozói segldeszközt tartalmaz. Ezek közül a leggyakrabban használtak az alábbiak:

Alkalmazásfejlsztési támogatás
szálak, aszinkron kommunikáció a szálak között, dinamikus modulbetöltés, memória-, pipe-, socket-, fájlkezelés, többszintű logolás
Fejlesztői segédeszközök
sztring-, dátum-, időkezelés, karakterkonverziós eszközök, parancssori paraméterek, XML, .ini, bookmark fájlok feldolgozása, időzítők, reguláris kifejezések
Adattípusok
láncolt listák, fák, asszociatív tömbök, szekvenciák, sorok, dinamikusan méretezhető tömbök
Itt is létezik C++ nyelvű változat, ahogy a GTK+ esetén is, ám itt a wrapper csak akkor készül el, ha a C++ standard könyvtára nem tartalmaz azonos funkcionalitású eszközt.

3 Első ablakunk

Ennyi bevezetés után épp ideje már némi kódot látnunk és azt górcső alá vennünk némiképp. Rögvest célszerű hozzátenni, hogy az alabbi -amúgy a gyakorlatban nem túl komoly hasznos-

ságú- példa teljes részletességgel történő kommentálása önmagában is egy cikksorozat témája lehetne. Szoritkozzunk most ennek okán csak a lényegre:

3.1 A kód

```
#include <gtk/gtk.h>

int main(int argc, char *argv[])
{
  GtkWidget *window;

  gtk_init (&argc, &argv);

  window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  gtk_widget_show  (window);

  gtk_main ();

  return 0;
}
```

```
#include <gtkmm.h>

int main(int argc, char *argv[])
{

  Gtk::Main kit(argc, argv);

  Gtk::Window window;


  Gtk::Main::run(window);

  return 0;
}
```

A fenti példaprogramok forrásfájljai és azok GTK+/gtkmm weblapjáról, az alábbi linkeken:

http://www.flosszine.org/sources/gtk_window.c

http://www.flosszine.org/sources/gtkmm_window.cc

http://library.gnome.org/devel/gtk-tutorial/stable/c39.html

http://www.gtkmm.org/docs/gtkmm-2.4/docs/tutorial/html/chapter-basics.html

3.2 Némi magyarázat

A lefordított programok természetesen ugyanazt az eredményt adják, de erről egy picikét később. Vegyük most szemügyre közelebbről a forráskódot amit a nagyfokó hasonlóság ellenére mutat eltéréseket is.

3.2.1 Projektek különbözőségéből adódó eltérések

A legtöbb nyílt forrású projekt esetén igaz, hogy a kódolás és kódszervezés során egy meghatározott konvenciót követnek, ám amint ez a korábbi példából is látszik, egy olyan méretű projekt esetén, mint a GNOME az egyes részterületeken lehetnek eltérések.

Forráskód formázása

A GTK+ fejlesztői a GNU coding standard irányelveit alkalmazzák, míg a gktmmm ettől minimális mértékben eltérő változat az alkalmaz.

Fejlécfájlok helye

Mint az későbbiekben megfigyelhető lesz a GTK+ headerek esetén a fájloknak van egy gtk prefixe, ugynakkor a gtkmm-nél ez nem használatos. Ez egyébiránt illeszkedik a nevezéktannál tapsztaltakkal.

3.2.2 Programozási nyelv okozta sajátosságok

Nevezéktan

A GTK+ minden saját makrót/függvényt GTK/gtk prefixszel lát el (GLib esetén ez pusztán csak egy kis/nagy g betű), sőt egy adott részterület -például egy widget- saját "névterülettel" is rendelkezhet, azaz újabb prefixet vezethet be. Ezeket egymástól, illetve a "valódi" funkciót jelölő nevektől _ (aláhuzás) jellel választják el. A C++ wrapper kódját olvasva láthatjuk, hogy kihasználva a kézenfekvő nyelvi lehetőséget, a prefixek szerepét a névterek veszik át, annyi különbséggel, hogy ezek neveiben csak az első betű nagy.

Nyelvhez kötődő módszerek

A minapéldában megfigyelhető (az azonos funkciójú sorok párbaállítása révén), hogy a C++ kód némileg egyszerűbb, hiszen az ablakunk létrehozásakor nem kell a window típusára (toplevel) vonatkozó paramétert megadnunk, azonos helyen lehet a változó deklarációja és első felhasználása. Külön is említésre méltó, bár a C++ prgramozók számára nem meglepő, hogy a main függvényből való kilépéskor a gtkmm változat windowja felszabadul, míg a GTK+ verzió esetén ez csak azért történik meg, mert magából a programból is kiléptünk egyúttal.

3.3 Vágyaink ablaka

3.3.1 Fordítás és linkelés

Az alábbi parancssorok segítségével fordíthatóak elkészült programjaink:

```
gcc gtk_window.c -o gtk_window `pkg-config --cflags --libs gtk+-2.0`
g++ gtkmm_window.cc -o gtkmm_window `pkg-config gtkmm-2.4 --cflags --libs`
```

Segítségünkre a pkg-config parancs van, hogy a gcc-nek a megfelelő paramétereket meg tudjuk adni. A `--cflags` paraméter hatására a fordításhoz, míg a `--libs` eredményeképp a linkeléshez szükséges opciókat kapjuk vissza. A parancs két ` (backtick) közé zárt, aminek hatására annak kimenete része lesz a fordító parancssorának, amivel pont az áhított hatást érjük el.

3.3.2 Futtatás

Ezek után már csak az örömteli pillanat van hátra, mikor is két különböző nyelven és függvénykönyvtárral lekódolt teljesen azonos funkciójú programunkat lefuttatjuk a `./gtk_window`, illetve a `./gtkmm_window` paranccsal.

3.3.3 Eredmény

Voilà! Túlvagyunk két ablak -vagyis egy Windows- lekódolásán. Ami eddig is közismert volt, újabb bizonyságot nyert, a Windows kódja lehet, hogy elegáns és egyszerű, ám az eredmény mégis hasznavehetetlen. Az utóbbin fogunk segíteni a következő részekben.

Pfeiffer Szilárd
2008. szeptember 10.

A cikkhez tartozó fórum címe:
http://www.flosszine.org/hello_window_01

Hello world!
Programozás Linux környezetben
2. rész

Hogyan adjunk át paramétereket a programunknak?

Az első részben megismerkedtünk a legeslegalapabb alapokkal, meg tudunk írni egy piciprogramot, le tudjuk fordítani és futtatni is tudjuk. Egy programtól azonban általában többet várunk el, mint hogy önmagában megcsináljon valamit „bedrótozott” adatokkal, programunkkal valamilyen módon tudatni kell, hogy valójában milyen bemenő és kimenő adatokkal, milyen választási lehetőségekkel szeretnénk ha lefutna. Ebben a részben a parancssori paraméterek átadásáról lesz szó.

Nézzük, hogy is nézett ki az előző rész programja:

```
#include <stdio.h>

int main (int argc, char* argv[]) {

    printf ("Hello world!\n");
    return (0);
} // main
```

A `main` függvény argumentum listájára azt mondtuk, hogy akkor és ott még ne foglalkozzunk vele. Nos ennek a foglalkozásnak az ideje jött most el. Mit is látunk ebben az argumentum listában? Egy egész számot és egy pointer tömböt méret megjelölés nélkül. Az `argc` értéke mutatja azt, hogy hány paramétert adtunk át a programnak. Használata egy picit trükkös, ugyanis ha nem adunk át egyet sem, akkor sem 0 az értéke hanem 1, ugyanis egy paraméter „láthatatlanul” mindenképp átadódik, ez pedig a programunk neve. Értelemszerűen ha egy paramétert átadunk parancssorból, akkor az `argc` értéke már 2 lesz. Friss tudományunkat rögtön próbáljuk is ki:

```
#include <stdio.h>
int main (int argc, char* argv[])
{
int i;
  printf ("A programunk neve: %s\n", argv[0]);
  printf ("A paraméterek száma: %d\n", argc-1);
  printf ("Az átadott paraméterek:\n");
  for (i = 1; i < argc; i++) {
    printf (" %s\n", argv[i]);
  } // for i
  return 0;
}
```

A program ugyanúgy fordítható és futtatható mint az első rész példaprogramja. Egy apróságot jegyeznék meg, sasszemű C kódereknek feltűnhet azonnal, hogy az egyik záró kapcsos zárójel után ott van a `// for i` megjegyzés. Ez nekem személyes szokásom, semmiféle kódolási konvencióban nincs benne, a ciklusok, `switch` utasítások, nagyobb blokkok, függvény deklarációk záró zárójele után egy C98 típusú megjegyzésben jelzem, hogy milyen blokkot is zár le az a zárójel. Még a múlt évezredben, amikor a számítógépekben a bug még igazi bogár betelepedését és a lábaival okozott rövidzárlat által előidézett hibát jelentette a processzorban,

az ALGOL-60 programozási nyelvben volt egy hasonló szintaktikai lehetőség, azt akkoriban roppant mód hasznosnak találtam és a zárójelek közti eltévelyedések mennyiségi csökkentése okán ezt a formáját a mai napig használom is. Visszatérve a mához, fordítsuk le a programot és paraméterek megadása mellett futtassuk is. Tulajdonképpen itt és most akár véget is érhetne ez a cikk, hiszen be tudtunk vinni paramétereket a programba. Ami miatt mégsem ér véget az azon egyszerű ok, hogy ezen paraméterek sok esetben teljesen egy kaptafára készülnek és működnek, azaz a problémakör szinte kiabál valami egységes kezelési módozatért, tán mégsem kéne felfedezni a spanyolviaszt minden egyes program megírásakor, a favágó munka helyett foglalkozhatunk a tényleges, érdemi alkotómunkával.

A parancssori paraméterek alapvetően két csoportba sorolhatók:

- opciók, más néven kapcsolók (options vagy flag néven ismertek angolul)
- egyebek, ide tartoznak pl. a fájl nevek

Az opcióknak két formája van:

A rövid (POSIX forma), kötőjellel „-” kezdődik és egy betű, például a `man -h`

A hosszú (GNU kiterjesztés), két kötőjellel „--” kezdődik és egy karakterlánc, szóközt értelemszerűen nem tartalmaz, hiszen az az argumentum lista elválasztó karaktere, példa rá a `man --help` mely működésében megegyezik a rövid formával.

A GNU parancssori konvenció szerint a programnak minden esetben ismernie kell a `--help` és a `--version` kapcsolókat. A parancssori kapcsolók feldolgozására a POSIX definiálta a `getopt()` függvényt, ennek GNU kiterjesztése a `getopt_long()`, mely a hosszú kapcsolókat is kezeli. Definícióik:

```
#include <unistd.h>

int getopt(int argc, char * const argv[],
           const char *optstring);
extern char *optarg;
extern int optind, opterr, optopt;


#define _GNU_SOURCE
#include <getopt.h>

int getopt_long(int argc, char * const argv[],
           const char *optstring,
           const struct option *longopts, int *longindex);

int getopt_long_only(int argc, char * const argv[],
           const char *optstring,
           const struct option *longopts, int *longindex);
```

Figyeljük meg, hogy a POSIX szerinti függvénydefiníciók a `unistd.h` helyen vannak, míg a GNU szerintiek a `getopt.h` helyen. Továbbá a GNU szerinti kiterjesztéshez (ha a Makefileban nem adtuk meg) definiálni kell a _GNU_SOURCE makrót. Mi a GNU szerinti kiterjesztést fogjuk használni, rögtön nézzük is meg egy példaprogramon.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include <errno.h>

const char *program_name;
const char *output_filename;

int o_flag;
int a_flag;

void print_usage (FILE *stream, int exit_code) {
  fprintf (stream, "Usage: %s options [input file(s)]\n", program_name);
  fprintf (stream,
   "  -h  --help             Display this usage information.\n"
   "  -a  --any-flag         Any option.\n"
   "  -o  --output filename  Write output to file (default: stdout).\n"
  );
  exit (exit_code);
} // print_usage

void process_options (int argc, char *argv[]) {
int next_option;
const char* const short_options = "hao:";
const struct option long_options[] = {
   { "help",       0, NULL, 'h' },
   { "any-flag",   0, NULL, 'a' },
   { "output",     1, NULL, 'o' },
   { NULL,         0, NULL, 0 }
};

  opterr = 0;
  do {
     next_option = getopt_long (argc, argv, short_options, long_options, NULL);
     switch (next_option) {
           case 'h':
                 print_usage (stdout, 0);
                 break; // h
           case 'a':
                 a_flag = 1;
                 fprintf (stderr, "a_flag SET.\n");
                 break; // s
           case 'o':
                 o_flag = 1;
                 fprintf (stderr, "o_flag SET.\n");
                 output_filename = optarg;
                 fprintf (stderr, "output_filename = %s\n", output_filename);
                 break; // o
           case '?':
                 print_usage (stderr, 1);
                 break; // ?
           case -1:
                 break; // -1
           default:
                 abort ();
     } // switch
  } while (next_option != -1);
} // process_options

int main(int argc, char *argv[])
{

  program_name = argv[0];
  output_filename = NULL;
  o_flag = 0;
  a_flag = 0;

  process_options (argc, argv);

/*
*
* Itt csinál valami hasznosat a program
*
*/

  return (0);
}
```

Mentsük el `floss_02.c` néven majd fordítsuk le a programot, egyelőre a már megszokott paranccsal:

```
gcc -Wall -o floss_02 floss_02.c
```

Futtassuk a `./floss_02` paranccsal. Majd ezek után próbáljuk meg különféle opciókkal, pl. `-h`, majd `-o valami-filenév`, érvénytelen opciókkal, hosszú opciókkal, satöbbi. Ha teljesen begyakoroltuk a program indítását a különféle opciókkal akkor pihenésképp nézzük meg, hogy valójában mit is csináltunk. A `main()` függvény roppant egyszerűen épül fel, kezdőértéket adunk a globális változóknak és meghívjuk a `process_options()` függvényt. Itt fogjuk az előbb ismertetett függvénnyel (`getopt_long()`) lebontani az opciólistát és egyenként értelmezni az így nyert elemeket. Először is meg kell mondanunk a függvénynek, hogy milyen opciókat keressen majd az opciók listájában:

```
const char* const short_options = "hao:";
const struct option long_options[] = {
     { "help",       0, NULL, 'h' },
     { "any-flag",   0, NULL, 'a' },
     { "output",     1, NULL, 'o' },
     { NULL,         0, NULL, 0 }
};
```

Egyszer ugye fel kell sorolnunk a rövid opciókat (melyek csak egyetlen betűből állnak), majd egy előre definiált struktúrából felépülő tömbben a hosszú paraméterlistát is megadjuk. A rövid opcióknál látunk egy kettőspontot (:) is. Ennek vajh mi a szerepe? Ezzel olyan opciót jelölünk, melynek **paramétere** is van, esetünkben egy fájl neve a `-o` opció paramétere. Természetesen több ilyen is lehet a rövid opciók listájában, ilyenkor több kettőspont is van a karakterek közt, például: `hao:xg:jkl`. Itt az -o és a `-g` kapcsolóknak van paramétere.

Nézzük a hosszú opciók rekordját, elég kézenfekvő a használata. Az első elem a hosszú opció neve, ez egy string literál. A második elem jelzi, hogy van-e paramétere ennek az opciónak. Ha `0` akkor nincs, ha `1` akkor van. [*1. házi feladat*] A harmadik paramétere vagy `NULL`, vagy pedig egy `int` típusú flag címe, ez esetben a `getopt_long()` függvény közvetlenül állítja a flaget, nekünk más dolgunk nincs [*2. házi feladat*]. Az utolsó elem pedig a *getopt_long()* függvény visszatérési értéke lesz, célszerűen az opció rövid nevét adjuk meg ehhez.

Az egyes opció kiértékelését egy ciklusban végezzük el, a `getopt_long()` visszatérési értékét egy `int` típusú változóba, jelen esetben a `next_option` változóba kérjük. Ezt cél-

szerűen egy `switch()` utasítással dolgozzuk fel, majd az egyes eseteknél a program belső működését befolyásoló flageket állíthatjuk be illetve az adott opcióhoz általunk elképzelt cselekménysort hajthatjuk végre.

A program használati utasításának kiíratását végző függvényhez sok megjegyzésünk nincs, általános jelleggel mutatnék rá, hogy a program által adott konzolos üzenetek kiíratásához az `stderr` kimeneti file-t célszerű használni.

Szemfüles Olvasónk észrevehet egy apró hiányosságot a programban. Ez a feladvány képezi az utolsó házi feladatot[*3. házi feladat*].

Felmerül(het) még egy kérdés az Olvasóban, hogy ha például olyan programot szeretnék írni, amely a `.c` kiterjesztésű file-okban számolja össze a karaktereket egy alkönyvtárban, akkor az ilyen programoknál használatos `*.c` típusú wildcardos fájl nevet hogyan dolgozom fel? Mert ugye maga a `*.c` meg fog jelenni az `argv[]` paraméterlistában, sőt akár opció paramétereként is megkaphatom, de valamit mégis kezdeni kellene vele.

Nos ennek a problémának az eleganс megoldásáról fog szólni a következő cikk. Addig is jó kódolást!

Házi feladatok:

[1] – Nézzünk utána a `libc` leírásban és a `getopt.h`-ban, hogy itt milyen értékek jöhetnek még szóba, ezeknek mi a funkciójuk és milyen előre definiált konstansok vannak ehhez.

[2] – Nézzünk utána ugyanott a flag közvetlen használatának.

[3] – Adjunk GNU-konform verziószámot a programunknak és ehhez rendeljünk opciót is, mind hosszút, mind rövidet.

Vomberg István

A cikkhez tartozó fórum címe:

http://www.flosszine.org/hello_world_programozas_linux_kornyezetben_02

Bash alapok

Noha manapság a Linux, és más alternatív rendszerek is rendelkeznek grafikus felhasználói felülettel, hozzám hasonlóan sokan szeretik a parancssoros felületet. Akkor pedig pláne jó, ha egy szerveren nem kapunk X-es felületet, csupán egy SSH konzolt. No meg ott a másik, ami miatt szeretem: sokszor egyszerűbb és gyorsabb megcsinálni az adott feladatot konzolból.

A bash-t szeretném bemutatni ebben a cikkben, azonban előre bocsátom, hogy nem vagyok bash mester, tehát könnyen lehet, hogy az én megoldásaimnál hatékonyabbakat is talál a kedves Olvasó.

A bash történetéről annyit mindenképp érdemes tudni, hogy 1978 környékén *Stephen Bourne* írt egy értelmezőt a Version 7 Unixhoz (sh), amelynek az „újjászületése” lenne a bash, (born-again shell). Mindezt Brian Fox követte el 1987-ben. 1990-ben Chet Ramey átvette a fejlesztést és mind a mai napig ő is vezeti. A program nem meglepő módon GNU/GPL licenc alatt érhető el.

Beállítások

A bash indulásakor számos helyen keres konfigurációs állományokat működésének beállításához. Ezek sorrendben a következők: `/etc/profile`, `~/.bash_profile`, `~/.bash_login`, `~/.profile`. Ez utóbbi három egyike, vagy mindegyike az aktuális felhasználó könyvtárában kap helyet. Ezek mellett fontos állományok még a `~/.bashrc` és a `~/.bash_logout`.
Azért tartottam célszerűnek kitérni ezen állományok létezésére, mert számos dolgot beállíthatunk bennük. Nálam például elég sok alias szerepel benne. Az aliasok felfoghatóak egyfajta parancsikonként. Egy ezek közül nálam:

```
alias alevt='alevt -vbi /dev/vbi0'
```

Ezzel közlöm a bash-al, hogy az „alevt” indításakor én az „alevt -vbi /dev/vbi0” utasítást kívánom meghívni. Ha emellett további opciókat szeretnék az alevt-nek átadni, akkor csak azt írom mögé. Például nálam az „alevt -geometry 40x25” parancsot a bash így értelmezi: „alevt -vbi /dev/vbi0 -geometry 40x25”. Ugye mennyivel rövidebb? (Az alevt egyébként egy remek teletext olvasóprogram Linux alá.)

Persze tetszőleges aliasokat készíthetünk, így például megoldható, hogy a `cp` parancsra az `rm` parancs fusson le. Mondjuk ez utóbbi már a jó ízlés határait sérti (aljas alias :-), de megvalósítható. Vagy például akár az is, hogy a *masol* parancs helyett a `cp` fusson le.

Pár hasznos dolog

A most következő példában feltételezek az Olvasóról minimális parancssori ismereteket, így például folyamatok kezelését és egyéb alapvető dolgokat.

Nézzük, mit lehet csinálni a bash-sel? Például lehet számoltatni:

```
for ((i=11;i<=20;i++)); do echo $i; done
```

Ez 11-től 20-ig kiírja a számokat külön sorba. Ha ugyanezt egy sorba szeretnénk kiíratni akkor hasonlóképp kell eljárnunk:

```
k=""; for ((i=11;i<=20;i++)); do k="$k $i"; done; echo $k
```

Ez annyiban trükkösebb, mint az előző, hogy egy ideiglenes változóba beírjuk a számot, mert különben az echo (alapértelmezés szerint) minden hívásakor soremelést hajt végre. Az ugye mindkét példán látható, hogy a sorokat pontosvesszővel kell zárni. Éppen ezért ha pontosvesszőt szeretnénk kiíratni, akkor backslash '\' jelet kell elé tennünk, mint sok más speciális karakter esetén, így például: $, \, &, ', „, `.

Az idézőjel és a normál aposztróf közötti különbséget jól érzékelteti az alábbi példa:

```
k=”valami”; echo '$k'; echo „$k”
```

Az elsőnél az úgynevezett teljes idézet, míg a második a részleges idézet.

Nézzünk egy kicsit bonyolultabbat:
30 és 40 között írassuk ki a 3-al osztható számokat:

```
for ((i=10;i<=13;i++)); do echo $((i*3)); done
```

Ez már egész szép, de nem nekünk kellene számolni, hanem a gépnek. Éppen ezért nézzünk olyan megoldást, ahol a 30-at és a 40-et adjuk meg:

```
for ((i=30;i<=40;i++)); do if [ $((i%3)) = "0" ]; then echo $i; fi; done
```

Itt már van egy feltétel vizsgálatunk is. Fontos, hogy a feltétel után van egy pontosvesszőnk, legalábbis ha egy sorba írjuk az egészet. Ennek elhagyása gyakori hiba. A másik fontos dolog az = jel. Ez szöveg szerinti egyezést ellenőriz, míg a `-eq` opció érték szerinti egyenlőséget vizsgál.

Az első igazi programunk

De mennyivel hasznosabb lenne ez a kis kódrészlet, ha nem lennének beledrótozva a paramé-

terek. Lássunk erre is egy példát:

```
#!/bin/bash
#Ez egy példaprogram

if [ $# -ne 3 ]
then
  echo A helyes paraméterezés: $0 x y z
  echo Ahol x a tól érték, y az ig érték, a z pedig a lépésköz.
else
  for ((i=$1;i<=$2;i++))
  do
    if [ $((i%$3)) -eq 0 ]
      then echo $i
    fi
  done
fi
```

Ugye valamivel átláthatóbb? Ami új ebben: a `$#` azt mondja meg, hogy hány paramétert kapunk. A `$0` mindig az aktuálisan futtatott fájl neve, míg a `$1..$n` rendre az első, második és n-edik paraméter. Érdemes tájékoztató üzenetet küldeni a helyes paraméterezésről, hiszen soha nem lehet tudni, hogy rajtunk kívül ki szeretné még a szcriptet futtatni. Hogy a programunk elindulhasson, ne felejtsük el futtathatóvá tenni azt a chmod paranccsal.

Természetesen –sok programnyelvhez hasonlóan– itt is megtalálható a `for` ciklus mellett a `while` ciklus is, mint ahogy az `if` feltételvizsgálatnál az `else` ág, valamint a `case`, mint esetvizsgálat.

A kézzelfogható haszon

A bash szkriptek egyik általam nagyon preferált felhasználási módja a tömeges képátméretezés. Az ImageMagick-el párosítva ugyanazon a számítógépen lényegesen gyorsabb, mint a kereskedelmi Adobe Photoshop. A legegyszerűbb paraméterezés például így nézhet ki:

```
for i in `ls p*.jpg`; do echo Most alakítom: $i; convert $i -resize
1024x768 -quality 75 m${i:1}; done
```

Nálam a fájlnevek *pXXXXXXX.jpg* formájúak. A fenti sor ezen állományokból készít 1024x768-as felbontású, 75%-os JPEG állományokat, melyek neve az *mXXXXXXX.jpg* mintát követi majd. A ${változó:mettől:hánykarakter} a substr függvényre hasonlít, mely számos programozási nyelvben megtalálható. Egyébként az ImageMagick segítségével nemcsak konvertálni, de például vízjelezni is tudunk egész gyorsan, erre például a composite programot javaslom.

Az is látható a példából, hogy a `for` ciklusunk nem csak egész számokkal tud dolgozni hanem halmazokkal is. Ha valaki programozott már PHP-ben, ott ez a `foreach`-nek felel meg. A másik érdekessége a *backtick* karakter, amely lehetővé teszi, hogy egy változónak ne mi adjunk értéket. Álljon erre példaként a következő sor:

```
datum=`date`; echo A mai dátum: $datum
```

Nagy mennyiségű videóállományt hogyan konvertálhatunk át a nekünk megfelelő formátumra?

A home könyvtáramban elhelyeztem egy szkriptet, melynek segítségével a videókat a telefonom számára is emészthető formára alakítom. A szkript neve viccesen: `egermozi.sh`, utalva a kínált felbontásra és a kijelző méretére. Lássuk a szkriptet:

```
#!/bin/bash
ffmpeg -i "${1}" -vcodec h263 -acodec libamr_wb -s qcif -ar 16000 -ab
23050 -ac 1 "/tmp/${1}.3gp"
```

A lustaság fél egészség, így nekem sem kell eztán a teljes parancssort beírni, elég csupán az alábbi:

```
egermozi.sh valami.avi
```

És ezután pár perc múlva már tölthetem is fel a `valami.avi.3gp` fájlt a telefonra. Viszont szerintem kicsit esztétikusabb ha az utolsó pont utáni részt levágjuk: `${1%.*}`, illetve hasonlóan pozitív ha levágjuk az állomány elérési útvonalát a kimenetnél a basename paranccsal, így elkerülhetjük a meglepetéseket.

A végleges szkriptünk tehát valahogy így néz majd ki:

```
#!/bin/bash
# tetszőleges mozi átalakítása 3gp formátumra
out=`basename $1`
out=${out%.*}”.3gp”
ffmpeg -i "${1}" -vcodec h263 -acodec libamr_wb -s qcif -ar 16000 -ab 23050
-ac 1 /tmp/$out
```

Remélem sikerült felkelteni a shell szkriptek iránti érdeklődést. Ugyanis azon túl, hogy félelmet kelt és kicsit fapadosnak tűnhet, hatalmas erőt adhat a felhasználó illetve a rendszeradminisztrátor kezébe.

A következő részekben kicsit még mélyebbre ásunk a bash bugyraiban, illetve szeretném bemutatni a sed, valamit az awk programokat is, melyek nem egyszer mentettek már meg több órányi gépies munkától, mely az említett két program segítségével csak néhány perc volt.

Ha valaki szereti a nyomtatott referenciakártyákat, akkor ez a segítségére lehet:
http://tldp.org/LDP/abs/html/refcards.html

E-Medve

A cikkhez tartozó fórum címe:
http://www.flosszine.org/bash_alapok_01

Az indián nyomában

Vitathatatlan, hogy az Apache a legnépszerűbb webkiszolgáló. Azonban trónkövetelők is vannak (nem, nem az IIS-re gondoltam), amelyek közül ebben a cikkben a lighttpd-t mutatom be.
A Unix környezetben már szinte egyeduralkodó Apache-csal szemben kisebb az erőforrásigénye, a készítői a biztonságra, gyorsaságra, a szabványoknak való megfelelésre és a rugalmasságra helyezték a hangsúlyt. Az alkalmazást olyan nagyobb, Web 2.0 szervezetek is használják, mint pl. a YouTube vagy a Wikipedia, ahol igen nagy terhelést kell kiszolgálni.

Töltsük le a http://www.lighttpd.net weboldalról a lighttpd legfrissebb verzióját, amely a cikk írásakor 1.4.20. Kicsomagolás után a telepítése a szokásos `./configure – make – make install` parancsokkal történik. Én a `./configure --prefix=/usr/local/lighttpd-1.4.20 --disable-ipv6 --with-zlib --with-openssl` opciókkal fordítottam, amely lehetővé teszi a https protokoll használatát, ill. az adatok tömörítve küldését (ha a kliens is támogatja), továbbá azt is megadtam, hogy most nem kérünk az IPv6-ból.
Jó, ha az `rrdtool` (http://www.rrdtool.org/) is telepítve van a gépünkön, mert a lighttpd a szokásos naplófájlok mellett egy RRD állományt is kezel, amelyből különféle statisztikát is gyárthatunk. Egy apró gond azért van vele: nem kezeli az újraindítást.

Konfiguráljunk!

A kiszolgálónak egyetlen konfigurációs állománya (`lighttpd.conf`) van, amelyet én a `/usr/local/etc` alá tettem.
Ahogyan az Apache esetében, a lighttpd-nél is használhatunk különféle modulokat, amelyeket itt a server.modules nevű részben kell megadni. Én a `mod_auth`, `mod_alias`, `mod_auth`, `mod_status`, `mod_fastcgi`, `mod_simple_vhost`, `mod_rrdtool` és `mod_accesslog` modulokat engedélyeztem, hogy a leggyakoribb Apache-szerű funkciókat használhassam.
Hasznos lehetőség az Apache esetében, hogy a „.ht” sztringgel kezdődő nevű állományokat alapból nem mutatja meg. Lighttpd esetén is szerettem volna, ha nem mutatja meg például a jelszóállományokat (.htpasswd), sem a backup-, sem pedig a .inc kiterjesztésű fájlokat. Ehhez az alábbi beállítás szükséges:

```
url.access-deny             = ( "~", ".inc", ".htpasswd" )
```

Virtuális hostok

Minden valamire való webkiszolgáló támogatja a virtuális hostokat, amelyekre a lighttpd két lehetőséget is biztosít: az egyiket a `mod_simple_vhost` modul, a másikat a `mod_mysql_vhost` segítségével. Az első az egyszerűbb, hiszen minden vhost konfigurációja a `lighttpd.conf`-ban szerepel, míg az utóbbi kényelmesebben menedzselhető (ehhez természetesen MySQL kiszolgálóra is szükségünk van). Fontos, hogy ne keverjük a két megoldást, csak az egyiket használhatjuk. Az alábbi példában két virtuális hostot definiálok, az utóbbinál egy aliast is megadtam.

```
$HTTP["host"] == "www.aaaa.fu" {
```

```
server.document-root = "/opt/www/data/www.aaaa.fu/"
}

$HTTP["host"] == "www.bbbb.fu" {
          server.document-root = "/opt/www/data/www.bbbb.fu/"
alias.url = ( "/pics/" => "/opt/www/data/pics/" )
}
```

Autentikáció

Természetes igény az, hogy bizonyos könyvtárakhoz csak megfelelő felhasználónév és jelszó megadása után lehessen hozzáférni. A lighttpd támogatja mind a basic, mind a digest metódusokat. A basic esetében a jelszó lehet titkosítás nélkül a jelszófájlban (plain), lehet titkosított - például DES, MD5, …- formában (htpasswd), ill. ún. „htdigest” - amely tartalmaz egy tartomány (realm) nevet is – továbbá LDAP kiszolgálón is tárolhatjuk a belépéshez szükséges adatokat. Az alábbi példában a htpasswd háttér használatára mutatok egy lehetséges megoldást. Jól látható, hogy itt nincs .htaccess állomány, ahol meg lehetne adni a belépéshez szükséges konfigurációt: a felhasználónevek és jelszavak kivételével minden a lighttpd.conf-ban szerepel.

```
auth.debug = 0
auth.backend               = "htpasswd"
auth.backend.htpasswd.userfile = "/opt/www/data/www.aaaa.fu/titkos/.htpasswd"

auth.require = ( "/titkos/" =>
          (
          "method"  => "basic",
          "realm"   => "top secret",
          "require" => "valid-user"
          )
         )
```

PHP

Manapság már elképzelhetetlen olyan kiszolgáló, amely nem ismeri a PHP-t. A lighttpd nem modulként (mint az Apache), hanem a FastCGI modul segítségével kezeli és futtatja a PHP scripteket. Ehhez az alábbi konfigurációt állítsuk be, amely azt definiálja, hogy a .php kiterjesztésű fájlokra vonatkozik, a localhost-on fognak futni (itt jegyzem meg, hogy akár másik gépen is futhatnak ezek a programok, illetve több gép esetén load-balance-ra is képes), a /tmp könyvtár alatt lesz a kommunikációs socket, ill. a php-cgi nevű binárist használja PHP értelmezőként. A PHP-t természetesen CLI-ként kell lefordítani.

```
fastcgi.server             = ( ".php" =>
          ( "localhost" =>
          (
          "socket" => "/tmp/php-fastcgi.socket",
          "bin-path" => "/usr/local/bin/php-cgi"
          )
         )
        )
```

URL átírás

Az Apache egyik nagyon hasznos funkciója a `mod_rewrite` modul, amely ugyanilyen néven a lighttpd esetében is elérhető, amely ugyan lényegesen egyszerűbb, mint az Apache modul, de mégis jól használható. Néhány egyszerűbb példa található a projekt honlapján: http://trac.lighttpd.net/trac/wiki/Docs%3AModRewrite .

SSL

Az alábbi példában a https://titkos.aaaa.fu/ nevű site-ot definiáljuk. A PEM fájl nem más valójában, mint a tanúsítvány és a kulcs egy állományban. Mivel a titkos kulcs is benne van, ügyeljünk a megfelelő jogosultság beállítására (0400 vagy 0600, illetve megfelelő tulajdonos).

```
$SERVER["socket"] == "1.2.3.4:443" {
ssl.engine                  = "enable"
ssl.pemfile                 = "/usr/local/etc/titkos.aaaa.fu.pem"
server.name                 = "titkos.aaaa.fu"
server.document-root        = "/opt/www/data/titkos.aaaa.fu/"
}
```

WebDAV

A lighttpd ismeri még a WebDAV protokollt is (amelyet fordításkor engedélyezni kell, és a `mod_webdav` modult bekapcsolni), a dokumentáció szerint az RFC 2518 egy nagyon minimális implementációját tartalmazza, azaz nincs az összes metódus megírva. A WebDAV többek között arra is jó, hogy megosztott naptár funkciót használjunk, mondjuk a Sunbird kliensoldali alkalmazással. Az alábbi konfigurációs részlet egy lehetséges példa ehhez.

```
$HTTP["url"] =~ "^/dav($|/)" {
webdav.activate = "enable"
webdav.is-readonly = "disable"
webdav.sqlite-db-name = "/opt/WebDAV/lighttpd.webdav_lock.db"
}
```

Záró szavak

A program lehetőséget ad arra is, hogy ne kelljen mindent a lighttpd.conf állományban tartanunk. Ehhez az `include_shell` direktívát használhatjuk fel, amelynek argumentuma egy futtatható program, ami a kívánt konfigurációs részletet kiírja a képernyőre (ill. a stdout-ra). Ezt felhasználhatjuk például a virtuális hostok karbantartására. Így tarthatjuk szöveges fájlban, sqlite3 adatbázisban, vagy bármi másban is az egyes virtuális hostok definícióit, azt a lighttpd induláskor beolvassa az említett futtatható program kimenetéről. A lighttpd Debian portja például arra használja ezt a funkciót, hogy a MIME típusokat a `/etc/mime.types` fájlból készíti el, a hivatkozása pedig így néz ki:

```
include_shell "/usr/share/lighttpd/create-mime.assign.pl"
```

Hasznos lehet, ha időnként meg tudjuk nézni a lighttpd státuszát, amelyet a `/server-status` címen érhetünk el. Ezen az oldalon kb. ugyanazokat az információkat nézhetjük meg, mint az Apache esetében. A használatához az alábbi sor szükséges:

```
status.status-url           = "/server-status"
```

Végül, ahogyan az Apache-hoz, úgy a lighttpd-hez is írhatunk saját modulokat. Ha egyszer sok időm lesz, lehet hogy írok egy olyan modult, amelynek segítségével MySQL táblából lehet autentikálni.

Sütő János

A cikkhez tartozó fórum címe:
http://www.flosszine.org/az_indian_nyomaban

A syslog-ng 3.0-ról dióhéjban

A magyar fejlesztésű syslog-ng (http://www.balabit.hu/network-security/syslog-ng/) a világ egyik legelterjedtebb rendszernaplózó alkalmazása, és a számos operációs rendszerben alapértelmezetten megtalálható syslogd leggyakoribb alternatívája. A támogatott platformok között szerepel a Linux, több BSD, a HP-UX, a Solaris, és az IBM AIX, de olvastam már Mac OSX-re fordított változatról is. A hamarosan megjelenő 3.0 kiadás kapcsán nézzük meg, hogyan is kell a syslog-ng-t beállítani, illetve mik a főbb újdonságok a 3.0-s kiadásban.

Elöljáróban: a cikk leadásakor még nem jelent meg a 3.0, ezért még nem találjátok meg a hivatalos letöltőoldalon (http://www.balabit.hu/network-security/syslog-ng/opensource-logging-system/upgrades/), de már nincs sok hátra :).

A syslog-ng konfigurálása

A syslog-ng a beállításait egy szöveges konfigurációs fájlból veszi, ami a legtöbb rendszeren (Linux, ...) a /etc/syslog-ng/syslog-ng.conf fájl. Egzotikusabb operációs rendszereken a syslog-ng.conf lakhelye legegyszerűbben a dokumentációból deríthető ki. Mivel most nem előregyártott konfigurációs fájlt fogunk használni, hanem nagy levegőt véve nulláról írunk egy sajátot, hozzuk létre ezt a fájlt kedvenc szövegszerkesztőnk segítségével. (Bátorságból kihívásokkal küszködők előtte mentsék el valahova az eredetit, vagy a syslog-ng indításakor az `-f <konfigfájl_elérési_útja>` kapcsolóval adják meg az új fájl elérési útját.)

```
sudo joe /etc/syslog-ng/syslog-ng.conf
```

Az első sorba gépeljük be az alábbiakat:

```
@version: 3.0
```

Erre a sorra azért van szükség, mert a konfiguráció szintaxisa az újdonságok miatt megváltozott a legutolsó nagyobb kiadás (2.1) óta, és így a syslog-ng már az új szintaxis szerint fogja tevékenységeinket értelmezni. A tapasztaltabb felhasználónak sem kell pánikba esniük, néhány figyelmeztető üzenetet után a régebbi konfigurációs fájlokkal is gond nélkül megy majd a 3.0.

A konfigurációs fájlban alapvetően kétféle dolgot kell csinálni:

1. Különböző objektumokat definiálni
2. Globális opciókat beállítani

A különböző objektumok közé tartoznak például a források (ahonnan az üzenetek jönnek), a célok (ahova az üzenetek mennek), illetve az üzenetek útját meghatározó útvonalak, az ún. log path-ok, amik megmondják, hogy a honnan érkező üzenetek hova menjenek. Globális opció például, hogy használjon-e a syslog-ng névfeloldást, vagy hogy a létrehozott naplófájloknak ki legyen a tulajdonosa. Az opciókat az options objektumban adhatjuk meg, pontosvesszővel elválasztva. Valahogy így:

```
options {use_dns (no); create_dirs(yes);};
```

Nem kötelező, de szokás az opciókat a konfigurációs fájl elején beállítani, így akkor is könnyen megtaláljuk őket, ha a naplózási szabályaink elburjánznak.

Az alapok

Elsőként definiáljunk egy forrást:

```
source s_internal {internal ();};
```

Itt láthatjuk, hogyan kell objektumot definiálnunk: megadjuk az objektum típusát (jelen esetben forrás, azaz source); nevét, ami bármi lehet, de érdemes az objektum típusára utaló előtaggal ellátni; kapcsos zárójelek közé zárva az objektum paramétereit; végül az egészet lezárjuk egy pontosvesszővel. Mivel ezt viszonylag könnyű elrontani, általában ajánlott a konfigurációs fájlt leellenőrizni a `--syntax-only` parancssori kapcsoló segítségével.

Ez egy speciális forrás, ami a syslog-ng saját üzeneteit tartalmazza. Vegyünk fel egy célt is, ahol szívesen látnánk ezeket az üzeneteket - például a `/var/log/mylogs.log` fájlban:

```
destination d_internal {file("/var/log/mylogs.log" flags(syslog-protocol));};
```

(A flags(syslog-protocol) rész igazából nem kötelező, ha kihagyjuk akkor a hagyományos, RFC3164 szerinti formátumnak megfelelően lesznek formázva az üzenetek, ha belerakjuk, akkor az új szabvány szerint.)

Ahhoz, hogy a definiált forrás üzenetei meg is érkezzenek a célként megadott fájlba, egy log objektummal kell összekötnünk őket:

```
log demo_log {
     source(s_internal); destination(d_internal);};
```

Mentsük el a konfigurációs fájlt, és töltsük újra a syslog-ng konfigurációs fájlját:

```
sudo /etc/init.d/syslog-ng reload
```

Láthatjuk, hogy a célként megadott `/var/log/mylogs.log` fájlt létrehozta a syslog-ng és a konfiguráció újratöltését jelző üzenet meg is jelent benne.

A syslog-ng belső üzeneteit naplózni feltétlenül hasznos de önmagában nem túl izgalmas, ezért vegyünk fel egy újabb forrást, ami az operációs rendszer üzeneteit gyűjti:

```
source s_devlog {unix-stream(/dev/log);};
```

Ha ezeket az üzeneteket is a korábban definiált `/var/log/mylogs.log` fájlba szeretnénk gyűjteni, adjuk hozzá ezt a forrást a logsorunkhoz:

```
log demo_log {
     source(s_internal); source(s_devlog); destination(d_internal);};
```

Vagy létrehozhatunk egy külön célt is, egy külön logsorral együtt:

```
destination d_devlog {file("/var/log/mydevlogs.log" flags(syslog-protocol));};
log demo_log {
     source(s_devlog); destination(d_devlog);};
```

Láthatjuk, hogy egy logsor több forrást (és több célt is) tartalmazhat. Ha sok forrásunk, célunk, és logsorunk van, esetleg még szűrőkkel is megbolondítjuk a helyzetet, akkor előfordulhat,

hogy egy üzenetet több helyre is elküldünk (a szűrőkről részletesen később), vagy véletlenül sehova sem. Az ilyen helyzetek kezelésére a logsoroknak két opcióját érdemes használni. Ha azt szeretnénk, hogy az egyik logsor által valahova elküldött üzenetek máshova már nem menjenek tovább (mert ott jó helyen van), akkor állítsuk be a final kapcsolót:

```
log demo_log {
     source(s_devlog); destination(d_devlog); flags(final);};
```

Ilyenkor az ide elküldött üzenetet a konfigurációs fájlban ezután következő logsorok már nem dolgozzák fel, vagyis fontos a logsorok sorrendje. A beérkező üzenetekre a syslog-ng sorban értékeli ki a logsorokat, ezért érdemes a final kapcsolót használó sorokat a lista elejére tenni.

A másik fontos kapcsoló a fallback, ami a "maradék" üzeneteket kapja el, pontosabban azokat, amiket a korábbi logsorok nem küldtek sehova. Tipikusan az utolsó logsornál érdemes ezt bekapcsolni.

```
log demo_log {
     source(s_logs); destination(d_lenyegtelen); flags(fallback);};
```

Távoli naplózás

Persze ne felejtsük el, hogy a syslog-ng távoli naplózásra lett kitalálva, vagyis a fő cél, hogy a klienseinkről a naplóüzeneteket egy másik gépre –a naplózószerverre– továbbítsuk. Ehhez két dolgot kell tennünk:

1. Felkészítenünk a naplózószerveren futó syslog-ng-t a hálózaton keresztül érkező naplóüzenetek fogadására
2. Utasítanunk a kliensen futó syslog-ng-t, hogy küldje el a beérkező üzeneteket a szervernek

A megoldás egyszerű: a kliensen vegyünk fel egy hálózati célt, a szerveren pedig egy hálózati forrást. A lenti példában a távoli naplózásra az IETF által kidolgozott új syslog protokollt használjuk (http://www.ietf.org/internet-drafts/draft-ietf-syslog-protocol-23.txt), ugyanis ennek több előnye is van a régi (RFC3164) protokollhoz képest: például alapértelmezetten ISO formátumú időpecsétet használ, támogatja a TCP hálózati protokollt is, és a réginél jóval kötöttebb üzenetformátumot használ. Lássuk.

A kliensoldali cél, és a hozzá tartozó logsor:

```
destination d_server {
     syslog("192.168.1.1" port(1999) transport("tcp"));};
log demo_log {
     source(s_internal); source(s_devlog); destination(d_server);};
```

Szerveroldalon kell egy hálózati forrás, ami a klienshez illeszkedő protokollon –jelen esetben TCP– érkező üzeneteket várja, valamint egy cél és egy logsor. Kezdetnek pakoljuk a beérkező üzeneteket a `/var/log/remote.log` fájlba.

```
source s_clients {syslog (port(1999) transport("tcp");};
destination d_logs {file("/var/log/remote.log");};
log {source(s_clients); destination(d_logs);};
```

Ha más -nem syslog-ng-t, vagy nem az új syslog protokollt használó- eszközökről is szeretnénk üzeneteket fogadni, akkor érdemes még két hálózati forrást is felvenni a TCP és UDP protokolloknak.

```
source s_old_clients {udp(); tcp();};
```

Ha a régi protokollnál nem az alapértelmezett 514-es portot használjuk, akkor meg kell adnunk a konkrét portszámot is:

```
source s_old_clients {udp(port(1999)); tcp(port(1999));};
```

Mivel a régi és az új syslog protokoll eltérő üzenetformátumot használ, érdemes külön fájlban tárolni a különféle üzeneteket.

```
destination d_rfc3164logs {file("/var/log/remote_rfc3164.log");};
log {source(s_old_clients); destination(d_rfc3164logs);};
```

Makrók

Ha több kliensről gyűjtjük az üzeneteket a szerverre, akkor kényelmetlen az egyes kliensektől beérkező üzeneteket azonos fájlban tárolni. Itt válnak hasznossá a syslog-ng makrói, melyekkel az üzenetek egy-egy adott részére vagy tulajdonságára hivatkozhatunk. Ha kliensenként külön fájlban szeretnénk látni a naplóüzeneteket, akkor a $HOST makrót kell használnunk:

```
destination d_logs {file("/var/log/$HOST.log")}
```

A syslog-ng számos makróval igyekszik életünket megkönnyíteni, így például a dátum különböző részeire is több makró létezik: a célban megadott útvonalat átalakítva `/var/log/$HOST/$DAY.log` kliensenként külön könyvtárban, napi bontásban tárolhatjuk naplóüzeneteinket. Itt fontos megjegyezni, hogy a könyvtárakat a syslog-ng automatikusan is létre tudja hozni ha engedélyezzük a create_dirs opciót:

```
options { create_dirs(yes);};
```

A makrók másik fontos felhasználási módja az üzenetek formátumának testreszabása. A makrókból sablonokat (template) rakhatunk össze, és minden célnak megadható, hogy milyen sablon szerint küldjük oda az üzeneteket. Az alapértelmezett üzenetformátum az "$ISODATE $HOST $MSGHDR$MSG\n", vagyis: a dátum ISO formátumban, időzónával együtt; az üzenetet küldő gép hosztneve vagy IP címe; az üzenetet küldő alkalmazás neve és PID-je; és maga az üzenet. (A $MSGHDR$MSG között direkt nincs szóköz, azt a $MSGHDR makró tartalmazza.) Ha nekünk például csak a kliens címére, az üzenet beérkezésének idejére, a küldő alkalmazás nevére és a naplóüzenetre van szükségünk, akkor az alábbi sablont kell használnunk:

```
destination d_file {
     file("/var/log/messages"
     template("$HOST $R_DATE $PROGRAM $MSG\n")); };
```

A sablonok természetesen tetszőleges szöveges részeket is tartalmazhatnak –mondjuk magyarázó szövegeket– tehát ha olvasmányosabbá szeretnénk tenni az előbbi sablont, akkor:

```
destination d_file {
     file("/var/log/messages"
     template(
          "Innen jött az üzenet:$HOST
          Ekkor kaptuk meg:$R_DATE
          Állítólag ez a program küldte:$PROGRAM
          Ez pedig maga az üzenet:$MSG\n")); };
```

Ha egy sablont több helyen is használni szeretnénk, akkor érdemes külön objektumként definiálnunk:

```
template t_demo_filetemplate {
     template("$ISODATE $HOST $MSG\n");};
destination d_file {
     file("/var/log/messages" template(t_demo_filetemplate)); };
```

Időnként előfordulhat, hogy egy üzenetnél egy makrónak nincs értéke, mert az üzenet nem, vagy hibásan tartalmazza az adott mezőt. Ennek tipikus példája, amikor hiányzik a küldő alkalmazás neve az üzenetből. A syslog-ng 3.0-ban azonban a makrókhoz alapértelmezett értékeket rendelhetünk, így például jelezhetjük az üzenetben, hogy ismeretlen alkalmazástól jött az üzenet:

```
destination d_file {
     file("/var/log/messages"
     template("$HOST $R_DATE ${PROGRAM:-ismeretlen} $MSG\n")); };
```

A makró alapértelmezett értékének beállításakor a ${MAKRÓNÉV:-érték} írásmódot kövessük, és ne feledkezzünk meg a kettőspont utáni kötőjelről.

Makrók terén még fontos megjegyezni, hogy a syslog-ng 3.0-ban az MSG és az MSGONLY makrók egymás szinonimái lettek, és csak az üzenet szövegét tartalmazzák – az MSG korábban az üzenetet küldő program nevét és PID-jét is tartalmazta. A programnév és a PID az MSGHDR makróval érhető el.

Naplóüzenetek tárolása adatbázisban

Manapság népszerű (majdhogynem trendi) dolog a naplóüzeneteket szöveges fájlok helyett adatbázisban (is) tárolni. A syslog-ng képes az üzeneteket közvetlenül adatbázisba küldeni: az adatbázist, a fájlhoz hasonlóan, egy külön célként kell definiálni, csak itt több paramétert kell beállítanunk. Hivatalosan az Oracle, MSSQL, MySQL, PostgreSQL, és SQLite adatbázisok támogatottak, de elvben bármelyik, a libdbi projekt (http://libdbi.sourceforge.net/) által támogatott adatbázis működésre bírható. **Adatbáziscél definiálásakor az alábbi paramétereket kell megadnunk:**

- az adatbázis típusa
- ha az adatbázis másik szerveren van, akkor a szerver IP címe és portja (megjegyzés: nem nagyon érdemes sok kliensről közvetlenül naplózni adatbázisba, praktikusabb, ha a kliensek az adatbázisszerveren futó syslog-ng-nek küldik az üzeneteiket, ami aztán továbbítja azokat az adatbázisnak)
- az adatbázis eléréséhez használt felhasználó neve és jelszava
- az adatbázis és a tábla neve (utóbbi makrózható)
- az oszlopok nevei, ahova az üzenet részeit beillesztjük
- az oszlopokba illesztendő üzenetrészekre hivatkozó makrók
- azon oszlopok listája, melyekhez az adatbázis készítsen indexet

Ha a megadott felhasználó rendelkezik a megfelelő jogosultságokkal, akkor a syslog-ng automatikusan létrehozza a táblát és a szükséges oszlopokat.

```
destination d_pgsql {
     sql(type("pgsql")
     host("naploszerver") username("syslog-ng") password("jelszo")
     database("naplok")
```

```
    table("messages_${HOST}_${R_YEAR}${R_MONTH}${R_DAY}")
    columns("datetime", "host", "program", "pid", "message")
    values("$R_DATE", "$HOST", "$PROGRAM", "$PID", "$MSGONLY")
    indexes("datetime", "host", "program", "pid", "message")); };
```

Üzenetek szűrése

Gyakori feladat a naplóüzenetek rendszerezése, illetve leválogatása valamilyen szempont szerint. Erre a syslog-ng beépített szűrőfüggvényei adnak lehetőséget. A szűrőket a logsorokban alkalmazhatjuk: a logsorba beérkező üzenetek közül csak a szűrőnek megfelelő üzeneteket továbbítja a megadott célnak, így könnyen kiválogathatjuk a számunkra fontos üzeneteket az üzenet tulajdonságai alapján: szűrhetünk pl. hosztnévre (host() szűrő), prioritásra (level() szűrő), de akár tetszőleges reguláris kifejezésre is (match() szűrő).

Például az alábbi szűrő csak a 192.168.1.1 gépről érkező, az üzenet szövegében a megadott kockacukor szót tartalmazó üzeneteket engedi át. A kockacukor helyett persze tetszőleges reguláris kifejezést is használhatunk, ha az adott feladat ezt kívánja.

```
filter pelda_filter {
    host("192.168.1.1") and message("kockacukor"); };
log demo_filteredlog{
    source(s1); filter(pelda_filter); destination(d1);};
```

Egy logsor több szűrőt is tartalmazhat. Ez akkor hasznos, ha a szűrőinket külön objektumként definiáljuk, hogy ugyanazt a szűrőt több logsorban is felhasználhassuk. Ebben az összes felsorolt szűrőnek teljesülnie kell az üzenetre (vagyis logikai ÉS kapcsolat van közöttük). Az előző példa működése tehát az alábbival megegyezik:

```
filter pelda_filter1 { host("192.168.1.1");};
filter pelda_filter2 { message("kockacukor");};
log demo_filteredlog {
    source(s1);
    filter(pelda_filter1); filter(pelda_filter2);
    destination(d1);};
```

Ami változás a syslog-ng korábbi változataihoz képest, a match("blabla") az eredeti (2.1-es) változatnak megfelelően működik ugyan, de lassabb, mivel meg kell formáznia az üzenet egy részét, mivel nem a "kész" értekben vizsgál. A `match("kockacukor" value("MESSAGE"))`; viszont ekvivalens a `message("kockacukorral")`-al mind sebességben, mind működésben. Másfelől viszont ezzel bármilyen üzenet-tulajdonságra lehet szűrni, azaz működőképes ez is:

```
match("kockacukor" value(".SDATA.meta.sequenceId"))
```

vagy akár bármilyen más parse-olt értékben való szűrés.

Parser használata

Az alapértelmezett szűrők és makrók a naplóüzenetek fejlécében, a metainformációkon alkalmazhatóak, az üzenetek szöveges részében nagyon korlátozottak a lehetőségeik. Ezen segít a syslog-ng 3.0 üzenetfeldolgozó (parser) modulja, ami valamilyen elválasztó karakter mentén külön mezőkre (név-érték párokra) bontja az üzenet szöveges részét. A mezőkhöz előtaggal (prefix) ellátott neveket kell rendelnünk, és ezeket a neveket makróként alkalmazhatjuk. Természetesen a parserek alkalmazásfüggőek, adott formátumot használó üzenetek feldolgozására

használhatjuk őket, emiatt legtöbbször szűrőket is érdemes mellettük használnunk. A parsereket a szűrőkhöz hasonlóan a logsorokban kell meghivatkozni. Fontos, hogy ha egy logsor több "kiegészítő objektumot" (szűrő, parser, átírási szabály) tartalmaz, akkor számít az objektumok sorrendje, ugyanis az objektumok által meghatározott műveleteket a syslog-ng a megadott sorrendben fogja végrehajtani. Röviden: nem mindegy, hogy előbb szűrünk, és a maradék üzenetet feldolgozzuk, vagy mindent feldolgozunk, és utána azok alapján szűrünk.

Parsert akkor alkalmazhatunk, ha a beérkező üzenetek valamilyen jól definiált formátumot használnak, és az egyes mezők (vagy legalább a mezők csoportjai) valamilyen azonosítható karaktersorozattal vannak elválasztva. Jó példa erre a vesszővel elválasztott lista. Alapértelmezésben a parser az üzenet szöveges részét ($MSG) dolgozza fel, de lehetőség van bemenetként makróval meghivatkozott értéket is megadnunk – például egy parser kimeneteként létrejövő makrót, így több lépcsőben az üzenet egyes részeit még tovább bonthatjuk. Nézzünk meg példaként egy formázott Apache üzenetet:

```
192.168.1.1 - - [10/Sep/2008:14:10:10 +0100] "GET /cgi-bin/examp-
le.cgi HTTP/1.1" 200 2708 "-" "curl/7.15.5 (i4 86-pc-linux-gnu) lib-
curl/7.15.5 OpenSSL/0.9.8c zlib/1.2.3 libidn/0.6.5" 2 example.balabit
```

Aki járatos az Apache naplóüzeneteinek olvasásában, az tudja, melyik mező micsoda, aki nem, attól most elnézést kérünk. A lényeg, hogy szóközzel elválasztott mezőkről van szó, és ahol egy mezőnek nincs értéke, ott ezt kötőjel karakter jelzi, tehát az üzenet mindig ugyanannyi mezőből áll. Definiáljunk hát egy parsert, elnevezve az egyes mezőket (előtag és nagybetű használata kötelező). Az elválasztó karaktert a `delimiter()` paraméterben, idézőjelek között tudjuk megadni. Fontos még, hogy maga az üzenet is tartalmazhat szóközöket egyes mezőiben, például az időpecsétben. Az ilyen elválasztókaraktereket akkor tudjuk kiszűrni, ha valamilyen idézőjelek között szerepel, ez a mi esetünkben most az időpecsétet keretező szögletes zárójel. Az ilyen idézőjeleket a `quote-pairs()` paraméterben, aposztrófok közé zárva sorolhatjuk fel. Mind a nyitó, mind a záró idézőjelet fel kell sorolni, mert azok nem feltétlen azonosak, tetszőlegesek lehetnek. Lássuk végre a parsert:

```
parser p_apache { csv-parser(
     columns(
     "APACHE.CLIENT_IP", "APACHE.IDENT_NAME", "APACHE.USER_NAME",
     "APACHE.TIMESTAMP", "APACHE.REQUEST_URL",
     "APACHE.REQUEST_STATUS", "APACHE.CONTENT_LENGTH",
     "APACHE.REFERER", "APACHE.USER_AGENT",
     "APACHE.PROCESS_TIME", "APACHE.SERVER_NAME")
     delimiters(" ")
     quote-pairs('""[]') );
};
```

Alkalmazzuk ezt egy logsorban. Ne felejtsük el, hogy a parserben definiált mezőket makróként alkalmazhatjuk, tehát például a honlapot meglátogató kliens IP címe alapján külön fájlba (vagy adatbázis táblázatba) is rendezhetjük az üzeneteket:

```
destination d_file {
     file("/var/log/${APACHE.CLIENT_IP:-noip}"); };
log {
     source(s_local); parser(p_apache); destination(d_file);}; };
```

Üzenetek átírása

A syslog-ng 3.0-val akár át is írhatjuk az üzenetek egyes részeit. Erre két módszer kínálkozik: az egyik egy adott mező beállítása fix értékre, a másik pedig a keresem-és-átírom (szinte sed). Az üzenetek módosításához átírási szabályokat (rewrite) definiálhatunk, amiket a szűrőkhöz és parserekhez hasonlóan alkalmazhatunk a logsorokban. Az első módszernél mindössze a mező új értékét és a mező nevét kell megadnunk. A mező tetszőleges makró lehet, például a hosztnév átírására az alábbi szabályt használhatjuk:

```
rewrite r_beallit {set("netuddki" value("HOST")); };
log { source(s1); rewrite (r_beallit); destination(d1);};
```

Ezzel az s1 forrásból érkező üzenet a d1 forrásban a netuddki hosztról érkezőnek fog tűnni. Az átírások helyét meghatározó makró lehet általunk parserben definiált makró is, úgyhogy ha jól strukturált üzeneteket kapunk, akkor azokat feldolgozva már elég sok mindent csinálhatunk az üzenettel – átírjuk, a belső tartalom alapján szűrjük és rendszerezzük, stb. Például az előző részben ismertetett Apache üzenetekből kitörölhetjük a kliens IP címét, ha szükséges:

```
rewrite r_noip {set("*.*.*.*"), value("APACHE.CLIENT_IP")};
log {
     source(s1);
     parser(p_apache); rewrite (r_noip);
     destination(d1);};
```

Ha nem konkrét mező értékét akarjuk módosítani, hanem úgy általában írnánk át valamit, akkor meg kell adnunk, hogy mit, mire, és hol szeretnénk megváltoztatni. A hol rész itt is egy makró, a mit és mire pedig valamilyen szöveg vagy reguláris kifejezés. Fontos, hogy alapértelmezésben a keresés érzékeny a kis- és nagybetűk közti különbségre, és csak az első találatot cseréli ki. Ezen az ingore-case és a global kapcsolókkal lehet segíteni. A következő példa az üzenet szöveges részét módosítja:

```
rewrite r_seekanddestroy {
     subst("ezt irjuk at", "erre", value("MESSAGE"));};
```

Többszintű logsorok

Az eddigiekből jól látszik, hogy a syslog-ng nagyon rugalmasan konfigurálható és szinte minden elemet újra fel lehet használni. Most lássuk, hogyan lehet a műveletek, pl. a szűrések eredményét több helyen is felhasználni. A korábbi változatokban erre nem volt lehetőség, vagyis ha például a bejövő üzeneteken alkalmaztunk egy szűrőt, hogy az eredményt elküldjük valahova, majd ezt a részeredményt tovább akartuk szűrni, akkor két külön logsort kellett írnunk, az egyikben egy, a másikban két szűrővel. Valahogy így:

```
log {
     source(s1); filter(pelda_filter1); destination(d1);};
log {
     source(s1);
     filter(pelda_filter1); filter(pelda_filter2);
     destination(d1);};
```

Mivel ilyenkor az első szűrést mindkét logsor elvégzi, ez sem átláthatóság, sem teljesítmény szempontjából nem előnyös. A syslog-ng 3.0 egyik újdonsága ezért a beágyazott, vagy többszintű logsorok bevezetése. A fenti példa így alakul át:

```
log { source(s_s1); filter(pelda_filter1); destination(d_file1);
     log {filter(pelda_filter2); destination(d_file2); }; };
```

Vagyis a megszokott logsor végén egy új logsort kezdünk, ami a szülőnek a kimenetét kapja meg, ezért nem tartalmazhat forrást. Egy logsor több beágyazott logsort is tartalmazhat, és a beágyazott logsoroknak is lehetnek beágyazott logsoraik, pl.:

```
log { source(s_s1); filter(pelda_filter1); destination(d_file1);
          log {filter(pelda_filter2); destination(d_file2); };
          log {filter(pelda_filter3); destination(d_file3);
               log {
                    parser(pelda_parser); filter(pelda_filter4);
                    rewrite(r_jolatirom); destination(d_file4); };
           };
};
```

Látható, hogy a d_file4 célba csak azok az üzenetek jutnak el, amikre a pelda_filter1, pelda_filter3, es pelda_filter4 szűrők egyaránt teljesülnek.

Ilyen esetekben persze fokozottan ajánlott valamilyen konzisztens módszer szerint tördelni a konfigurációt, hogy könnyen olvasható maradjon. Természetesen a beágyazott logsorok nem csak szűrőket, hanem például parsereket is tartalmazhatnak, így egész bonyolult feldolgozás is lehetséges némi munka árán.

Merre tovább

A fentiekben megpróbáltam érzékeltetni, mennyire sok mindenre használható a syslog-ng, kicsit belekapva a legérdekesebb tulajdonságaiba. A syslog-ng iránt komolyabban érdeklődőknek ajánlom figyelmébe a hivatalos dokumentációt, ami az alkalmazást fejlesztő BalaBit cég honlapján érhető el: http://www.balabit.hu/support/documentation/. Bár a cikkben nem tértünk ki rá, a syslog-ng 3.0 nyílt forrású változata is képes az üzeneteket TLS-titkosított és kölcsönösen autentikált kapcsolaton keresztül továbbítani. Ennek a beüzemelése (tanúsítványok generálása és telepítése, stb.) kicsit munkás, de igény esetén egy későbbi cikkben részletesen kifejthetjük.

Ha a program használata során nehézségbe ütköznétek, érdemes kicsit keresgélni a syslog-ng levlista archívumában (http://marc.info/?l=syslog-ng&r=1&w=2), vagy egyszerűen feldobni a kérdést a listára (https://lists.balabit.hu/mailman/listinfo/syslog-ng).
A dokumentációval kapcsolatban fontos megjegyezni, hogy az egyben a syslog-ng kereskedelmi változatának a dokumentációja is, így abban olyan tulajdonságokról is szó esik, amelyek nem részei a nyílt forrású változatnak. Ilyen például a merevlemezes pufferelés, a titkosított logfájl használat. A két változat közötti különbségeket a dokumentáció elején megtaláljátok. Ami még jó hír, hogy a kereskedelmi változat syslog-ng Agent for Windows alkalmazásának (Windows eventlog üzeneteket syslog-ng szerverre továbbító program) hamarosan lesz ingyenes -bár nem nyílt forrású változata- is.

Addig is kellemes naplopást (izé... naplózást)!

U.i.: Segítségéért és a példák ellenőrzéséért külön köszönet illeti Tusa Viktort.

Fekete Róbert

A cikkhez tartozó fórum címe:
http://www.flosszine.org/syslog-ng_3_0_diohejban

Webmin

Négyezer-hatszáz év múlva, az aktuális nyelvészkongresszus soron következő előadásának témája, akár egy égett szélű papírlap is lehet, melyen csak ennyi látszik: Webmin. Számukra nyújtunk egy kis segítséget, (persze ehhez meg kell találniuk ezt a szöveget kinyomtatott formában, aminek a lehetőségét grönlandi futurológusok erősen vitatják).

Webmin -> web admin, ráadásul még csak web sem kell hozzá.

A fejlesztők által létrehozott rendszer kiváló vitageneráló, hiszen egyszerre lehet kétes és ugyanakkor hasznos. Kétes, mivel a rendszergazdák egy része nem hajlandó a konzolon történő script és parancs íráson kívül más rendszeradminisztrációs eszközt használni, ezért erre is gyanakodva néznek, ugyanakkor hasznos, mert bizonyos esetekben megkönnyítheti a rendszergazda munkáját. Megint csak kétes, mivel felesleges biztonsági kockázatot jelent, ugyanakkor egy védett hálózaton, több szervert felügyelő jó munkásembereknek mégis nyújthat némi segítséget.

Eme bő lére eresztett, fikcionális bevezető után nézzük a lényeget.

A szoftver a webmin.com-ról tölthető le. Itt a bevezetőben foglaltaknál kevésbé körülményesen, viszont sokkal érthetőbben elmagyarázzák, hogy a szóban forgó alkotás, a Unix config állományok kézi szerkesztését hivatott kiváltani.

Ugyanitt a Usermin és a Virtualmin is megtalálható. Előbbi integrálva van ugyan a Webmin-be, de szükség lehet rá, hogy külön kezelhető legyen a webmail, levélszűrők, stb., csak erre korlátozott jogkörrel, így felhasználóinknak is lehetővé tehető, hogy adminisztrálják saját levelező rendszerük beállításait. Utóbbival egy interfészen keresztül kezelhetünk több virtuális szervert, de ez talán nem a legpontosabb megfogalmazás. Itt most azonban csak a Webmin-ről esik szó.

A Webmin jelenleg az 1.430-as verziónál tart. Hogy mi mindent konfigurálhatunk vele, abba inkább itt ne menjünk bele, hiszen ez a program saját weboldalán megtekinthető. Kedvcsinálónak néhány tétel az alaprendszer buherálásán túl: *Apache, Bind, BSD Firewall, Bacula, Frox, Jabber, MySQL, Postfix, PostgreSQL, Qmail, Samba, Squid*, stb. Ennél sokkal több matatásra van lehetőségünk, a standard modullista megtekinthető itt: http://www.webmin.com/standard.html. Látható, hogy még a Solaris Zónákat is elbarmolhatjuk a rendszer segítségével.

A szabványos modulokon kívül rengeteg más modul létezik hozzá, sőt senki sem akadályozza meg a felhasználót abban, hogy maga is létrehozzon új modulokat.

Nehéz lenne nem támogatott rendszert találni hozzá, de mint említettem a támogatás el is készíthető, így a rendszer valóban széleskörűen használható. A jelenlegi helyzet: http://www.webmin.com/support.html. Itt többek között azt is leírják, hogy jelenleg a legszélesebb körű támogatás Solaris, Linux (Red Hat) és FreeBSD rendszerekhez érhető el.

A hivatalos oldalról letölthető RPM, DEB, TAR, Solaris Package formában, és ugyanitt találhatjuk a fejlesztői verziókat, valamint a mások által készített modulokat. Még Windows verziót is találhatunk, bár ez korlátozott, ami talán nem is baj.

Érdemes tudni, hogy biztonsági okokból néha kimaradhat egy-egy disztribúcióból, de aki mégis úgy dönt, hogy szüksége lenne rá, többféleképpen is telepítheti.

Erre egy példa az alábbiakban:

Debian 4.0

A sourceforge.net-ről letölthetőek az aktuális csomagok, jelenleg 1.430.

```
wget http://prdownloads.sourceforge.net/webadmin/webmin_1.430_all.deb
```

A függőségeket telepíteni kell:

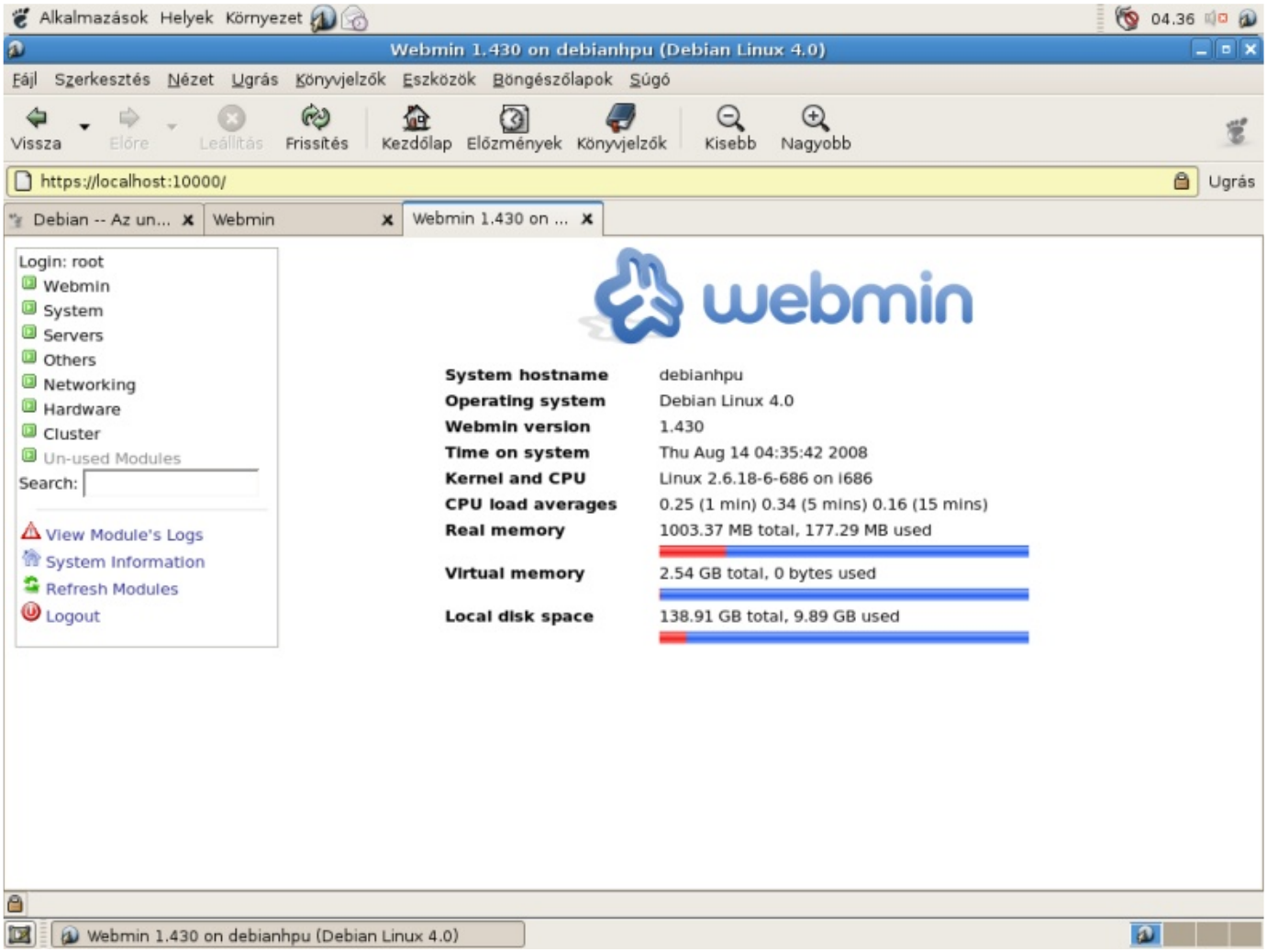
```
apt-get install libnet-ssleay-perl libauthen-pam-perl libio-pty-perl libmd5-perl
```

Csomag telepítése:

```
dpkg -i webmin_1.430_all.deb
```

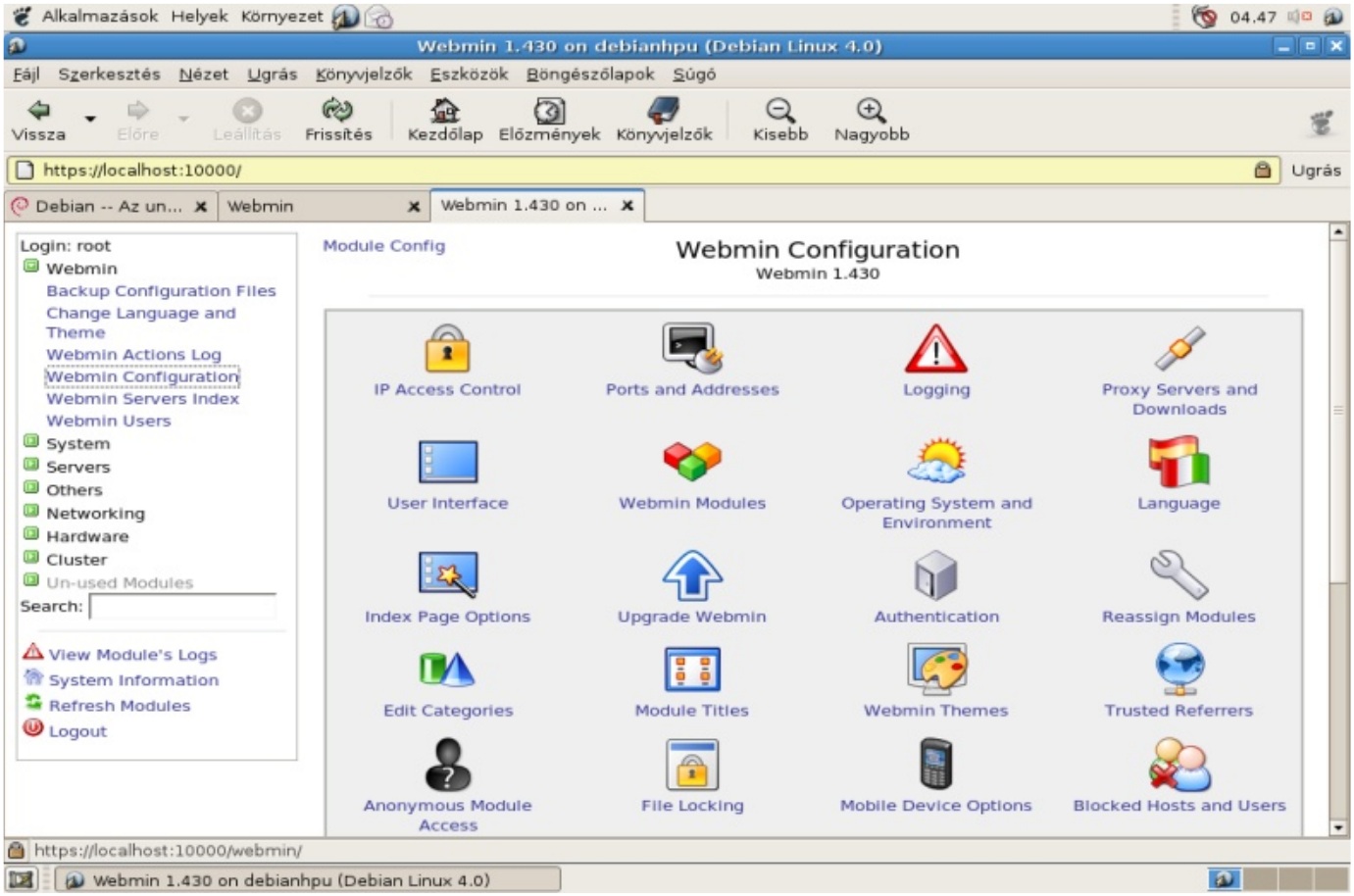
Ezután a rendszer az adott gépen, böngészőből elérhető, az alábbi címen:
https://localhost:10000
Amennyiben telepítésre került a Usermin, az a https://localhost:20000 címen érhető el.
Belépni rootként tudunk. Érdemes rögtön kreálni egy másik usert, hogy ne rootként kártékonykodjunk a gépünkön, persze figyelve a beállításokra, mert csak ahhoz lesz jogunk, amit beállítunk magunknak.
A rendszerbe való belépés után rengeteg lehetőség fogad bennünket. Szeretném felhívni a figyelmet, hogy a Webmint egyesek kezdőknek is ajánlják, de szerintem amíg valaki az állományrendszerben és a szkriptekben, config állományokban valamelyest nem ismeri ki magát, a rendszer csak arra szolgál, hogy kényelmessé tegye a szerverszolgáltatások tönkretételét. Bizonyos tapasztalat mellett viszont remek segédeszköz válhat belőle.
Előnyös, hogy nem köti meg a kezünket, amellett, hogy a grafikus felületen szinte számtalan beállítási lehetőséget találhatunk, nyugodtan barkácsolhatunk tovább kézzel is, mikor mihez lesz kedvünk, ez egyébként valószínűleg tevékenységfüggő lesz, hisz van amit az egyik módon találunk kényelmesnek, van amit a másikon.

Belépés után az alábbi képernyő látható:



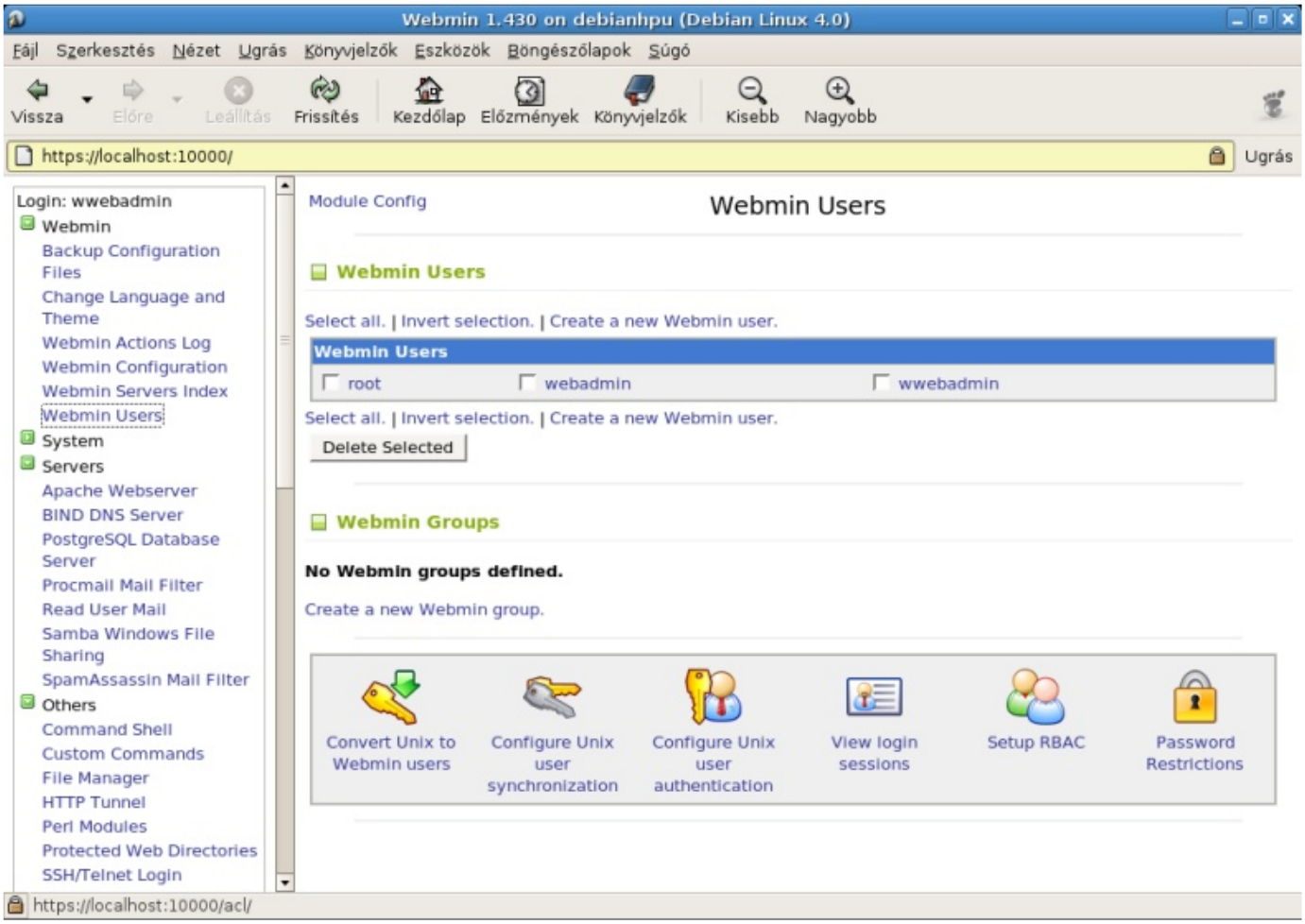
Középen a rendszerinformáció jelenik meg, ennek is van külön menüpontja, hogy ha valahol másutt ténfergünk, könnyen megjeleníthessük a gépre vonatkozó adatokat.
Bal oldalon található a menü panel, legfelül látjuk, milyen felhasználóként dolgozunk, amivel néha nem árt szembesülni. Ez alatt az érdemi tevékenységet takaró menüpontok következnek.

Webmin: itt a Webmin rendszerrel kapcsolatos beállításokkal találkozhatunk. Az első máris egy igen hasznos szolgáltatás, a rendszerállományok mentésének mikéntjét és hogyanját állíthatjuk be, nemcsak az alaprendszerét, de a Webmin, és a használt szerverek állományai is megadhatók.
Backup Configuration Files: készíthetünk azonnali, valamint ütemezett mentést, és a visszaállítás dicsőséges tevékenységét is innen kezdeményezhetjük.
A Backup menüpont alatt egy kis csicsaközpont következik, nemcsak a használt nyelvet válthatjuk át például koreaira vagy japánra, de a megjelenést is variálhatjuk. (Míg nagyon sok nekünk egzotikus nyelvre sikerült beállítani (japán, kínai, koreai, sőt fárszi/perzsa), addig klingonul és héberül vagy arabul nem volt hajlandó megjelenni, bár ezek is megtalálhatóak a listában).
A **Change Language and Theme** menüpont után a **Webmin Actions Log** tétel következik. Itt eléggé széleskörű lehetőségünk lesz arra, hogy a logolás nemes tevékenységét egy böngészőn keresztül beállítsuk. Ugyanerről a felületről később kereshetünk a log állományokban Webmin user, modul és dátum szűrőkkel is.
A **Webmin Configuration** bejegyzés olyan kövér lehetőségeket takar, hogy az erről írt oldalak kitennének egy kötetet, nemcsak a Háború és Béke sorozatban, de a Mao Összes Beszédeinek gyűjteményes kiadásában is.
Részletes felsorolás helyett álljon itt egy képernyőkép, talán olvasható lesz:



Mint látható (!?!), az IP hozzáférések szabályozásán felül, a portokat és címeket is kontrollálhatjuk, a már említett nyelvek és témák variálásán felül. Témák letöltésében, és a felhasználói felület részletes testreszabásában is maradandót alkothatunk itt (legalábbis a következő módosításig).

Sok egyéb mellett a modulok kezelésére is itt nyílik lehetőség.
Kénytelenek vagyunk továbblépni, ha el akarunk jutni a végéig, így következik a **Webmin Servers Index** menüpont, ahol a Webmin szerverek után tapogatózhatunk, sőt ezt a tevékenységet automatikussá is tehetjük itt.
Rögtön ezután következik a **Webmin Users** menüpont. (Az első bejelentkezés után érdemes azonnal idejönni, és megtenni, amit megkövetel a józan ész). Itt lehet (és érdemes) új felhasználó(ka)t létrehozni, netán törölni (a listázás automatikusan megtörténik).



Az adminisztráció megkönnyítésére csoportokat is képezhetünk. Ha van csoportunk, a Unix felhasználóinkat konvertálhatjuk Webmin userekké, beállíthatjuk a szinkronizációt a kétféle felhasználói kör között, sőt az alaprendszer felhasználóinak hozzáférését is szabályozhatjuk. Megnézhetjük, hogy kik vannak bejelentkezve, sőt a kapcsolódó logokba is betekinthetünk. Ezen felül szabályozhatjuk a csoportra vonatkozó jelszavak erősségét is.
Ugyanitt van lehetőségük a Solarist használóknak -de csak nekik- (ne legyen túl régi) az RBAC (Role Based Access Control) beállítások matatására is.

Nagy nehezen elérkeztünk a következő főbb menüponthoz, melynek neve System.
System: nem meglepő módon itt a rendszerrel kapcsolatos dolgokat szabályozhatjuk.
Elsőként a **Bootup and Shutdown** menüponttal találkozunk.

Jó, ha figyelünk, mert az itt elvégzett változtatások erősen kihatnak rendszerünk működésére. A Module Config most is az adott modul beállításaira vonatkozik, ez alatt egy csinos kis táblázat található (felette és alatta „Create a new bootup and shutdown action.” mondattal, amire kattintva létrehozhatjuk saját gyártmányú akciónkat).
A táblázat első oszlopa egy pipatároló, ahol is az adott tevékenységet kapcsolhatjuk be, vagy ki. Utána az akció (ez sok minden lehet) nevére kattintva kapunk egy új képernyőt, ahol az adott bejegyzést szerkeszthetjük. Itt adható meg, az adott akció neve, az azt végrehajtó szkript

(ezt szerkeszthetjük is), valamint az, hogy a boot folyamat során végrehajtódjon-e? (ennek ténye a következő oszlopban fog látszani a táblázatban).
A szerkesztő doboz alatti gombokkal kimenthetjük, elindíthatjuk, leállíthatjuk és törölhetjük az akciót. A törlés gombbal vigyázzunk, mert kiválasztása és lenyomása esetén alaphelyzetben minden további kérdezősködés nélkül törli az adott akciót a táblázatból. Ez a gombsor akciónként más és más lehet, ezért az előbb leírtakon kívül néha találkozhatunk az újraindítás, vagy a folyamat állapotának megnézése lehetőségekkel is.
A táblázat következő oszlopa jelzi, hogy az adott akció végrehajtódik-e a rendszer elindulásakor. Az utolsó oszlopot az adott akció rövid ismertetése foglalja el (ha van ilyen).
A táblázat alatt található néhány -az akciókkal kapcsolatos- indítási, beállítási lehetőség, plusz a rendszer újraindítása, leállítása, futási szint beállítása, stb.

Az ezután következő **Change Passwords** menüpont eléggé egyértelmű, bár mivel minden rendszerszintű felhasználót is megjelenít, nem érdemes a beállításaikat eszetlenül bolygatni.

A **Disk and Network Filesystems** menüpont szintén kifejezetten buherálásra csábító dolog, de ne dőljünk be könnyen az itt felkínált lehetőségeknek, hacsak nem pályázunk a Pokoli Operátor babérjaira. A menüpont kiválasztása után kilistázásra kerülnek a felcsatolt tárterületek, főbb jellemzőikkel együtt.
A felcsatolt terület megnevezésére kattintva (ahol lehet) megkapjuk a beállítási képernyőt, itt a mountolt terület beállításait szerkeszthetjük.
A területek felsorolása előtt és után új területeket, eszközöket csatolhatunk a rendszerbe.

A **Filesytem Backup** menüpontban az időzített mentés adatait állíthatjuk be.

Az **Initial System Bootup** menüpontban a rendszer inicializálásának folyamatát ~~állíthatjuk be~~ állíthatjuk el, ez sem kifejezetten kezdőknek való tevékenység.

A következő, **Log File Rotation** menüpont elég világos és hasznos, a naplózott eseményeket nyilvántartó állományok megőrzését, felülírását állíthatjuk be, a kívánalmaknak megfelelően (legyen elég információ, de ne teljen be túl hamar a tárterület).

Császárunk ezután adja nekünk a MIME típusú programok beállítási lehetőségeit a **MIME Type Programs** menüpont képében. MIME (Multipurpose Internet Mail Extensions), azaz Sokcélú Internetes Levélbővítések. A MIME egy olyan specifikáció, amely leírja, hogyan továbbíthatók különböző formátumú adatok az Internet levelezőszabványaival, pl.: multimédia állományok.
A rendelkezésre álló táblázat segítségével az ilyen típusú programokra/állományokra vonatkozó beállításokat végezhetjük el itt, szükség esetén új MIME típust is felvehetünk.

Következik a **PAM Authentication** menüpont. Ez nem a *gugli(M)* által elsőként felhozott P A M oldal látogatóinak azonosítását szolgáló beállításokat takarja, hanem sokkal inkább a (Pluggable Authentication Modules), Csatlakoztatható Azonosítási Modulok beállítását.
Az azonosítási szolgáltatások széleskörű beállításán túl új PAM szolgáltatást is vehetünk fel.

Túl sok haszontalansággal eddig sem találkoztunk a Webmin felületi boncolgatása során, de most egy, a hasznosságán túl, rendkívül kényelmes szolgáltatással találkozhatunk, a **Running Processes** menüpontra bökve. Folyamatokat listázni karakteres képernyőn is egyszerű, de itt a táblázatos formában megjelenő, különböző módon listázható (PID, User, Memory, CPU) folyamatok között talán könnyebben megtaláljuk a bennünket érdeklő processt, vagy rákereshetünk a Search ablakban, de újat is indíthatunk a Run képernyő beviteli mezőit megfelelően kitöltve.

Ezután a **Scheduled Commands** menüpont jön, ami magáért beszél, így én hallgatok, annyit azért hozzáfűzve, hogy a beviteli ablakban minden szükséges információ megadható, ami a parancs időzített futásához szükséges (a gépet azért arra az időre se kapcsoljuk ki).

A **Scheduled Cron Jobs** csak kicsit lesz érdekesebb bejegyzés, ennél már ismét kapunk egy kis táblázatot, ahol a Cron feladatütemező által elvégzendő munkálkodást felügyelhetjük.
A már megszokott módon -a bejegyzésekre kattintva- szerkeszthetjük a paramétereket, létrehozhatunk új bejegyzést, szabályozhatjuk a felhasználók hozzáférését az adott szkripthez, stb.

A **Software Packages** menüpontban nem meglepő módon a csomagkezelés hasznos tevékenységéhez kapunk segítséget. Nézegethetjük szeretett csomagjainkat, újakat tehetünk fel, stb.
A **System Documentation** menüpont segítségével a man oldalak, és egyéb dokumentációs lehetőségek tárházában keresgélhetünk különböző szempontokat megadva.

A **System Logs** pontnál a rendszer által létrehozott naplóállományok beállításait módosíthatjuk, hozhatunk létre újakat, és nézegethetjük a már meglévők tartalmát.

A **Users and Groups** menüpont a felhasználók és csoportok beállításainak szerkesztésére szolgál. A megjelenő táblázatból megtudhatjuk a szükséges információkat, az adott felhasználóra, vagy csoportra kattintva pedig szerkeszthetjük azok paramétereit. Ugyanebben a menüpontban hozhatunk létre, törölhetünk, stb. felhasználókat és csoportokat.

Ezzel végére is értünk a System menüpontnak, következik a Servers bejegyzés.
Servers: Logikus lenne, hogy itt a szerverekkel kapcsolatos lehetőségek legyenek összegyűjtve. Lássuk! Szerencsére így van, ahogy elvárható, rákattintva a menüpontra az almenüben felsorolásra kerülnek a rendelkezésre álló szerverek. (Nem minden szoftveres megvalósításnál egyértelmű az ilyesmi, hallottam, hogy vannak olyanok, akik nap mint nap kénytelenek együtt élni azzal a szörnyűséggel, hogy a kilépés menüpontot a start menügomb alatt találhatják meg, van olyan is, aki ezért még mindig nem tudta kikapcsolni a gépét).
A szerverek alatt felsorolt menüpontok száma és felirata erősen függ a rendszerben elérhető szerverszolgáltatásoktól. Ugyanígy az almenü pontok tartalma is az adott szolgáltató szervertől függ.
Mindenesetre az **Apache Webserver**, vagy a **Samba Windows File Sharing** menüpontra kattintva, mindig az adott szerver beállításait fogjuk megtalálni, ami elég megnyugtató.

Mivel nem könyv íródik, csak egy rövid összefoglaló, nem venném sorra a különböző szerverek beállítási lehetőségeit, maradt még így is elég menüpont.
A következő vastag betűs, a sokatmondó **Others** nevet kapta a keresztségben. A félreértések elkerülése végett mások helyett egyebeknek fordítanám, így az egyebek menüpontba került sok egyéb dolog az eddigieken kívül. Mik is ezek?

Legfelül a parancskagyló, a **Command Shell** lakik. Szerencsére nem a kagyló fog nekünk parancsolgatni, hanem mi parancsolgatunk a kagylónak, vagyis kiadhatunk a Webminből egy futtatható parancsot, amit a parancsértelmező fog értelmezni (ez a dolga) és végrehajtani.

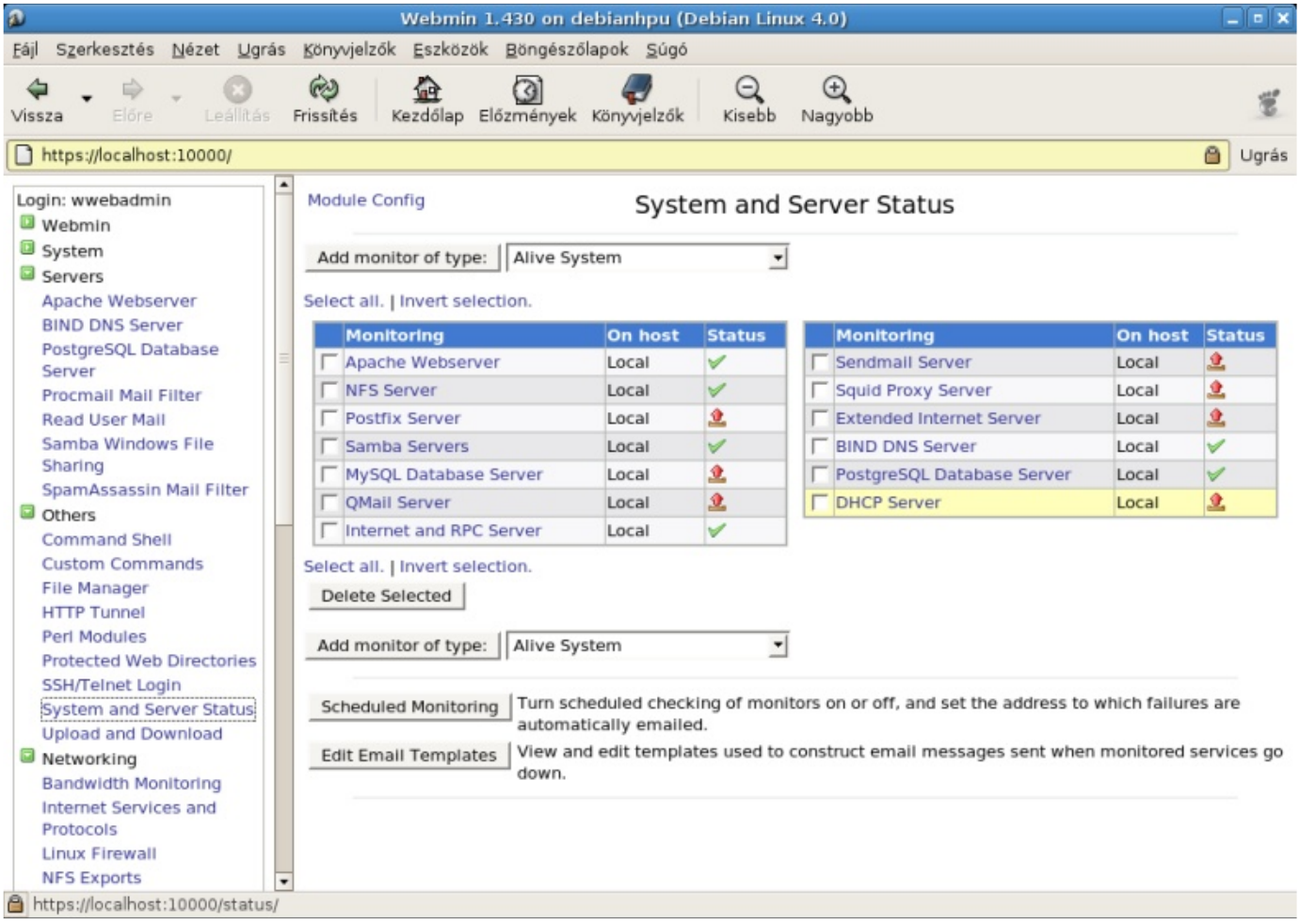
Comandante Hostigar után következik a **Custom Commands** menüpont. Itt parancsokat hozhatunk létre, a létrehozott parancsnál beállíthatjuk azt is, hogy elérhető legyen-e a Usermin alól?

A következő lehetőség a **File Manager** nevet viseli, ahol valóban lehetőségünk nyílik az

állományok menedzselésére (böngészőtől függően néha kiabál, hogy töltsük le a megfelelő plugint).
Ezután a **HTTP Tunnel** menüpont jön, itt az előzetes beállításoknak megfelelően az URL beírása után létrejön az alagút, ha szerencsénk van, kicsit gyorsabban, mint ha egy löszfalban próbálnánk meg furkálni.

Következik a **Perl Modules**. Lehetőség van a meglévő modulok megnézésére, rájuk kattintva részletesebb információt kapunk (az almodulokról is), itt leszedhetjük őket, és újak feltelepítésére is lehetőségünk van.

A **Protected Web Directories** menüpontban a védett könyvtár adatait adhatjuk meg és a felhasználói hozzáféréseket is szabályozhatjuk.
Ha fut Telnet (biztonsági megfontolások miatt nem ajánlott) vagy SSH szerver, azok beállításait az **SSH/Telnet Login** menüpontban adhatjuk meg.
Alatta a **System and Server Status** menüpont található, (lehetne felette is, de az almenük ábécés sorrendben következnek egymás után, így alá került) itt különböző szempontok szerint monitorozhatjuk rendszerünket, és a szerverszolgáltatásokat.



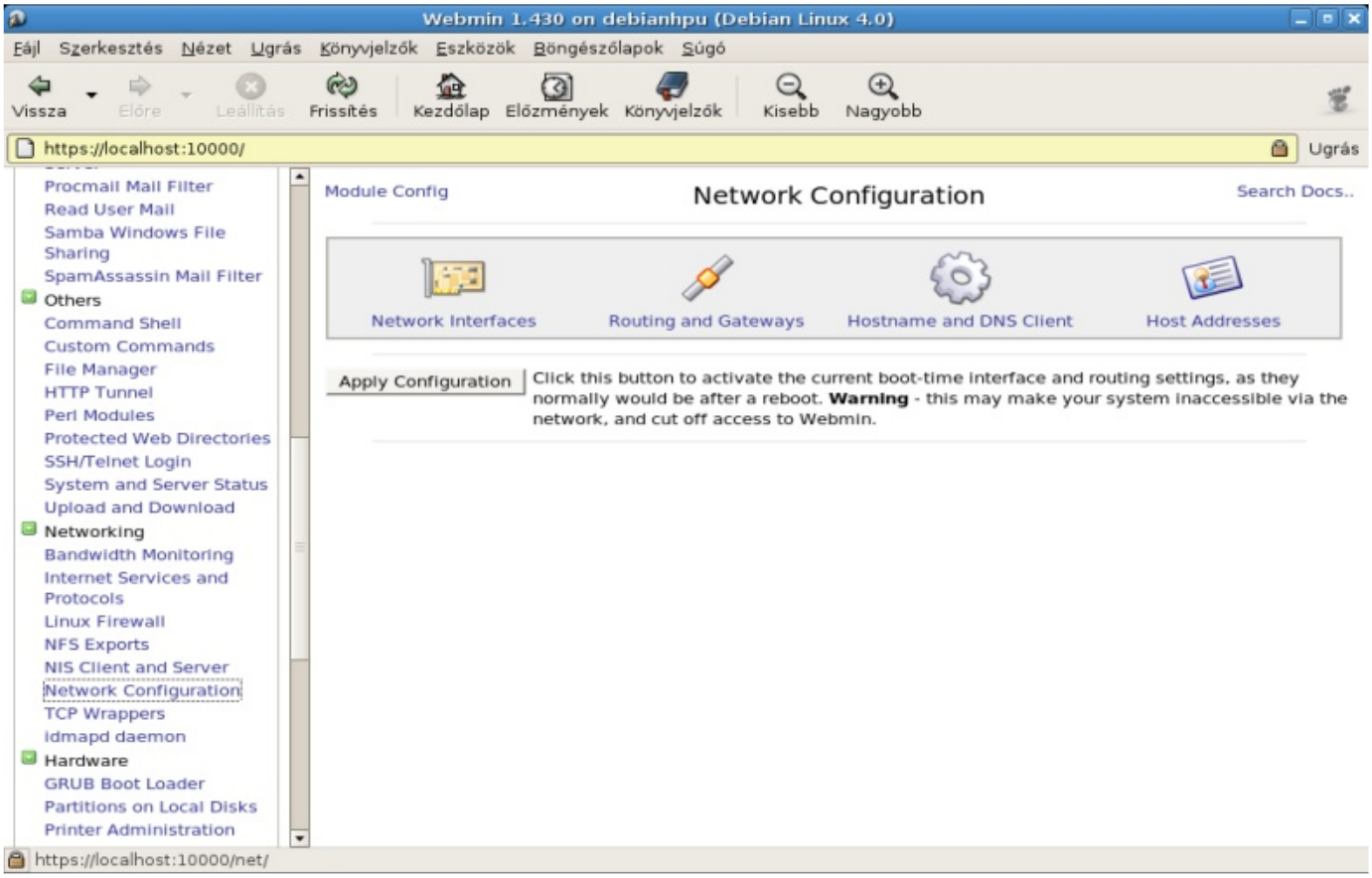
A táblázatban látható, hogy az adott szerver installálva van-e, ha igen, lokálisan fut-e, stb. A nevére kattintva további információkat és beállítási lehetőségeket kapunk. A monitorozás lehet ütemezett, és különböző szempontok szerinti.
Ebben a szekcióban az utolsó bejegyzés az **Upload and Download**. Itt a fel és letöltésekkel kapcsolatos paramétereket szabályozhatjuk.
Következő főbb menüpont a **Networking**, ahol néhány hálózattal kapcsolatos beállítási lehetőséget találhatunk. A **Bandwidth Monitoring** a sávszélesség ellenőrzésében lesz segítségünkre.

A következő, **Internet Services and Protocols** menüpontban láthatjuk a futó Internet szolgáltatásokat, és beállíthatjuk a rájuk vonatkozó paramétereket, valamint új szolgáltatásokat is létrehozhatunk és indíthatunk. Ezen felül itt találjuk az RPC-kre (Remote Procedure Call), Távoli Eljáráshívásokra vonatkozó beállítási lehetőségeket is.

Továbblépve következik a **Linux Firewall** menüpont. Ez már önmagában is egy nagyon szép, és kiterjedt téma. Az alapok beállítása aránylag egyszerű, de a finomságok túlbonyolítása esetén hamar az elveszettség érzése kerítheti hatalmába a delikvenst. A képernyőn a szokásos lehetőségek közül választhatunk (packet filtering, packet alteration, network address translation). Például a csomagszűrést választva, a bejövő, kimenő és továbbított csomagokra érvényes szabályokat beállíthatjuk, és új szabályokat hozhatunk létre.

Rögtön alatta találjuk az **NFS Exports** menüpontot, ahol a Need for Speed könyvelőprogramban elért eredményeinket oszthatjuk meg másokkal. Ja, nem! Itt az NFS Network File Systemmel kapcsolatos beállításokat variálhatjuk.

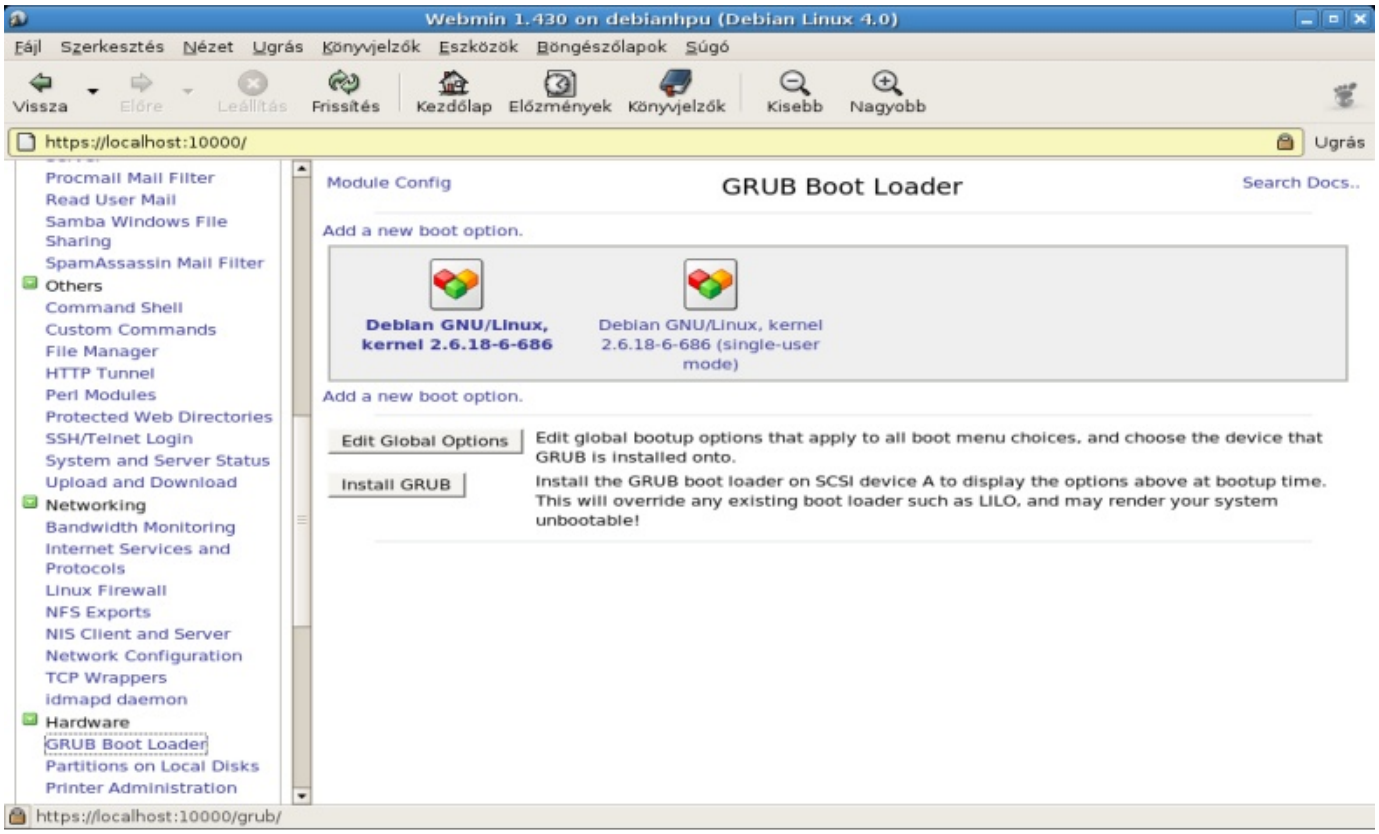
Következik a **NIS Client and Server** menüpont. Ez meg mi? ***FLOSSzine Totó következik:***
X. Norton Internet Security.
1. Jugoszlávia (vagyis ami még megmaradt belőle) második legnagyobb városa.
2. Hálózati Információs Rendszer (Network Information Service), azaz NIS.
A kettes nyert! ;) Vagyis itt állíthatjuk be a Sun (már megint) által kifejlesztett szolgáltatás paramétereit.
Rögtön utána található a **Network Configuration** menüpont ahol a hálózati beállításokkal kísérletezhetünk mint az egyszeri vegyész gyerek, rögtön a labor felrobbantása előtt.



A modul konfigurálásán kívül foglalkozhatunk a hálózati kártyákkal, ismerkedhetünk a route-olás rejtelmeivel, a host és DNS beállításokat, és a host címeket is megváltoztathatjuk, ha akarjuk.
TCP Wrappers a következő bejegyzés a menüben. Ennek a modulnak a használatával hálózati szolgáltatásokat felügyelhetünk. Alapból a szabályok az /etc/hosts.allow és az /etc/hosts.deny állományokba kerülnek. Hasznos először mindent tiltani, majd a szükséges dolgokat engedélyezni.

Most egy démoni menüpont következik, **idmapd daemon**.
Ez az NFS 4-esben megjelent új (daemonként futó) szolgáltatás, ami a felhasználói azonosításból adódó problémák megoldásában segít. Az UID-ek és GID-ek névre fordítását végzi el szükség szerint. A menüpontban néhány ezzel kapcsolatos beállítást végezhetünk el.

Szépen haladunk, eljutottunk a **Hardware** főmenüponthoz és még a századik oldalnál sem tartunk. A hardver mindjárt egy szoftver beállításával kezdődik, a **GRUB Boot Loader** almenüpont, a GRUB rendszerbetöltő beállításait taglalja és elősegíti, hogy könnyen és gyorsan elérjük a mindenki által rettegett állapotot, mikor is a rendszer nem indul. (Hiába szép a grafikus felület, itt minden apró ténykedésünknek súlyos következménye lehet a rendszer épsége szempontjából).
Persze kellő fegyelmezettséggel és önmegtartóztatással kivitelezhető az is, hogy nem nyúlunk a működő beállításokhoz, és akkor – legalábbis miattunk – nem lesz semmi baj.



Egyébként nagyon egyszerű és kényelmes módon matathatunk a GRUB menüben, és ha biztosak vagyunk a dolgunkban, a végén rábökhetünk az Install GRUB gombra. Amennyiben LILO, vagy más bootmanager volt a gépünkön, ne aggódjunk, ez majd szépen felülírja, nem törődve annak káros következményeivel.

Ezután a **Partitions on Local Disks**, sokatmondó nevű menüpont következik. Itt bizony a meglévő partícióinkat szerkeszthetjük, törölhetjük (többnyire), és újakat hozhatunk létre. Ez is sokkal vidámabb móka itt, mint egy spártai karakteres felületen, persze a hatása ugyanolyan rettentő lehet.

A következő, **Printer Administration** menüpont a rendszer működése szempontjából kevésbé súlyos, de nyomtatási igények felmerülése esetén igen hasznos beállítási lehetőségeket rejt, itt a nyomtatók beállítását végezhetjük el. **System Time** menüpont. Itt állíthatjuk be a rendszeridőt.
Ezzel a hardver résznek vége, következik a **Cluster** főmenüpont.
Ez a fürtözéssel kapcsolatos beállítások lelőhelye. Amíg nem veszünk fel menedzselhető szerver(eke)t, addig túl sok mindent nem csinálhatunk itt. Viszont nincs szükség nagy

gépparkra, mivel virtuális szervereket is kezelhetünk, így megnézhetjük a beállítási lehetőségeket, ha saját szerverünket felvesszük.
Ezzel a menüponttal is elég sokáig el lehet szórakozni, végül is értelme csak akkor van, ha valóban van több fizikai szerverünk, amiket fürtözve, egyszerre tudunk menedzselni.
Ebben az esetben a jelszavak kezelése, az állományok másolása, a **Cron Jobs**-ok kezelése, a shell parancsok kiadása, a szoftvercsomagok menedzselése mind könnyebbé válik. Usermin és Webmin szervereket egyaránt felvehetünk, és a csoportok és felhasználók kezelését is fürtben végezhetjük.

Az **Un-used Modules** menüpont megjeleníti azokat a modulokat, amelyek kezelhetőek lennének a Webminnel, de nincsenek a gépünkön, vagy a konfigurációjuk nem megfelelő. Ezen segíthetünk, ha telepítjük az adott modult, vagy megfelelően konfiguráljuk, ha már fent van, és a Webmin mégis ide sorolja.
Ez alatt található a kereső doboz, ahol a Webmin rendszerben kereshetünk.

A **View Module`s Logs** menüponttal megnézhetjük a modulok logállományait, ha engedélyezve van a naplózási tevékenység.

A **System Information** menüpontról már volt szó, ezt láthatjuk a Webminbe való bejelentkezés után, de bármelyik képernyőről ideugorhatunk a menüpont használatával.

Maradt még a **Refresh Modules**, ahol megnézhetjük, milyen modulok vannak használatban, és melyek nem, valamint a **Logout**, amely menüponttal kiléphetünk a rendszerből, ha befejeztük áldásos tevékenységünket.

Még egy hasznos lehetőségünk van, **amennyiben nem akarjuk telepíteni a Webmint, akkor kipróbálhatjuk a www.webmin.com -ról**. A *Demo and Screenshots* menüpont használatával beléphetünk egy Webmin demo szerverre, itt a *root* és *demo* párost használhatjuk, közvetlen link:
http://webmin-demo.virtualmin.com - Ez sajnos nem mindig működik, vagy egy Virtualmin Pro (ez a Virtualmin prémium verziója) szerverre, szintén root/demo, közvetlen link:
http://virtualmin-demo.virtualmin.com - Most ez is megadta magát, tegnap még működött, biztos elkattintgatták magukat :).

Ez a kis rövid összefoglaló nem pótolhat egy a Webmin rendszerről szóló könyvet (nem is ez volt a célja), de remélhetőleg felkelti az érdeklődését azoknak, akik egy ilyen segédeszközt szeretnének használni. Magyar nyelvű Webmin könyv tudtommal nem létezik, ha valaki tud ilyet kérem jelezze, viszont az angol nyelven megjelent *The Book of Webmin (Joe Cooper)*, és a *Managing Linux Systems with Webmin: System Administration and Module Development (Jamie Cameron)* könyvek valamelyike, vagy a *Webmin kompakt (H. Uman)* zsebkönyv hasznos lehet. Ezekről a könyvekről véleményt mondani nem tudok, mivel nem állnak a rendelkezésemre, viszont akár használt, de jó állapotban, igen olcsón megrendelhetőek az Interneten keresztül.
A Webmin saját dokumentációja néhol elég hiányos, de a dokumentáció készítésben lehet segíteni a fejlesztőknek.

Legvégül annyit szeretnék hozzáfűzni, hogy a Webmin nagyszerű segédeszköz lehet, de teljesen kezdőknek mégsem ajánlható a használata, hiszen az alapfogalmak ismerete nélkül a webes felületen való kattintgatásnak súlyos következményei lehetnek. A Webmin a rendszerállományokat közvetlen módon éri el, adatbázisokat és egyéb, nem szabványos adattárolási módokat nem használ. A rendszerrel való ismerkedést ugyanakkor jól szolgálja, hiszen a modulok tanulmányozása során olyan beállítási lehetőségekre is bukkanhatunk, amiknek talán nem is tudtunk a létezéséről. Annyit azért nem szabad elfelejteni, hogy minden lehetőséget a Webmin sem bocsát rendelkezésünkre, és a kényelem oltárán, néha kicsit a hatékonyságból kell feláldoznunk.

Szőke József

A cikkhez tartozó fórum címe:
http://www.flosszine.org/webmin

Mi van ott?

A nagy sikerű nmap programot mindenki ismeri, amelynek segítségével megtudhatjuk, hogy egy adott gépen milyen portok vannak nyitva, ill. szűrt állapotban. Az *amap* és *vmap* hasonló funkciót látnak el, csak éppen az alkalmazásrétegben működnek, és nem a TCP/UDP szintjén.

A két program a THC (The Hacker's Choice) web oldaláról tölthető le:
http://freeworld.thc.org/releases.php
Telepítésük a szokásos `./configure; make; su -c 'make install'` parancsokkal történik.

Az amap teljesen hasonló módon működik, mint az nmap: a `connect()` függvény segítségével kapcsolódik a távoli géphez. Ez azt is jelenti, hogy csak TCP alapú szolgáltatások detektálására képes, UDP alapúakat, pl. DNS, nem tud.

Nézzünk meg egy példát, és adjuk ki az alábbi parancsot!

```
amap -d -o 1a.txt -b 1.2.3.4 21 22 80 110 443 995
```

Ennek hatására az alábbihoz hasonló eredményt kapunk.

```
amap v5.2 (www.thc.org/thc-amap) started at 2008-10-19 15:25:17 - MAPPING mode
Protocol on 1.2.3.4:22/tcp matches ssh - banner: SSH-2.0-OpenSSH_5.1\r\n
Identified response from 1.2.3.4:22/tcp (by trigger http):
0000:  5353 482d 322e 302d 4f70 656e 5353 485f    [ SSH-2.0-OpenSSH_ ]
0010:  352e 310d 0a                               [ 5.1..            ]
Protocol on 1.2.3.4:22/tcp matches ssh-openssh - banner: SSH-2.0-OpenSSH_5.1\r\n
Identified response from 1.2.3.4:22/tcp (by trigger http):
0000:  5353 482d 322e 302d 4f70 656e 5353 485f    [ SSH-2.0-OpenSSH_ ]
0010:  352e 310d 0a                               [ 5.1..            ]
Protocol on 1.2.3.4:80/tcp matches http - banner: HTTP/1.0 200 OK\r\nConnection
close\r\nContent-type text/html\r\nContent-Length 3697\r\nDate Sun, 19 Oct 2008
132518 GMT\r\nServer lighttpd/1.4.20\r\n\r\n<!
DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http//www.w3.org/TR/xhtml1/DTD/xhtml
Identified response from 1.2.3.4:80/tcp (by trigger http):
0000:  4854 5450 2f31 2e30 2032 3030 204f 4b0d    [ HTTP/1.0 200 OK. ]
0010:  0a43 6f6e 6e65 6374 696f 6e3a 2063 6c6f    [ .Connection: clo ]
0020:  7365 0d0a 436f 6e74 656e 742d 7479 7065    [ se..Content-type ]
0030:  3a20 7465 7874 2f68 746d 6c0d 0a43 6f6e    [ : text/html..Con ]
....
```

Jól látható, hogy a program megpróbál a megadott portokhoz kapcsolódni, és a kapott eredményt kiírta a képernyőre. A program nem egyszerűen csak kiírja pl. az ssh bannert, hanem rengeteg - érvénytelen – adatot küld az adott portra, és az ezekre kapott válasz alapján próbálja meg kitalálni, hogy milyen, ill. milyen jellegű alkalmazás van a túloldalon, pl. webszerver, Oracle kiszolgáló, stb.

Egy sniffer programmal vizsgálva a hálózai forgalmat, az amap - többek között - az alábbi adatokat küldte el a 22-es portra (bal oldalt a csomagok tartalma hexadecimális formában):

```
47 45 54 20 2f 20 48 54    54 50 2f 31 2e 30 0d 0a    GET / HTTP/1.0..
0d 0a                                                 ..
80 80 01 03 01 00 57 00    00 00 20 00 00 16 00 00    ......W... .....
13 00 00 0a 07 00 c0 00    00 66 00 00 07 00 00 05    .........f......
00 00 04 05 00 80 03 00    80 01 00 80 08 00 80 00    ................
00 65 00 00 64 00 00 63    00 00 62 00 00 61 00 00    .e..d..c..b..a..
60 00 00 15 00 00 12 00    00 09 06 00 40 00 00 14    `...........@...
00 00 11 00 00 08 00 00    06 00 00 03 04 00 80 02    ................
00 80 63 b9 b9 19 c0 2b    ae 90 74 4c 73 eb 8b cf    ..c....+..tLs...
d8 55 ea d0 69 82 1b ef    23 c3 39 9b 8e b2 49 3c    .U..i...#.9...I<
5a 79                                                 Zy
03 00 00 0b 06 e0 00 00    00 00 00                    ...........
79 08 00 00 00 01 00 00    00 00 00 00 20 43 4b 41    y........... CKA
41 41 41 41 41 41 41 41    41 41 41 41 41 41 41 41    AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41    41 41 41 41 41 00 00 21    AAAAAAAAAAAAA..!
00 01                                                 ..
81 00 00 44 20 45 42 45    4e 45 42 46 41 43 41 43    ...D EBENEBFACAC
41 43 41 43 41 43 41 43    41 43 41 43 41 43 41 43    ACACACACACACACAC
41 43 41 43 41 00 20 45    42 45 4e 45 42 46 41 43    ACACA. EBENEBFAC
41 43 41 43 41 43 41 43    41 43 41 43 41 43 41 43    ACACACACACACACAC
41 43 41 43 41 43 41 00                               ACACACA.

00 00 00 85 ff 53 4d 42    72 00 00 00 00 18 53 c8    .....SMBr.....S.
00 00 00 00 00 00 00 00    00 00 00 00 00 00 ff fe    ................
00 00 00 00 00 62 00 02    50 43 20 4e 45 54 57 4f    .....b..PC NETWO
52 4b 20 50 52 4f 47 52    41 4d 20 31 2e 30 00 02    RK PROGRAM 1.0..
4c 41 4e 4d 41 4e 31 2e    30 00 02 57 69 6e 64 6f    LANMAN1.0..Windo
77 73 20 66 6f 72 20 57    6f 72 6b 67 72 6f 75 70    ws for Workgroup
73 20 33 2e 31 61 00 02    4c 4d 31 2e 32 58 30 30    s 3.1a..LM1.2X00
32 00 02 4c 41 4e 4d 41    4e 32 2e 31 00 02 4e 54    2..LANMAN2.1..NT
20 4c 4d 20 30 2e 31 32    00                          LM 0.12.

48 45 4c 4f 20 41 4d 41    50 0d 0a                    HELO AMAP..

55 53 45 52 20 41 4d 41    50 0d 0a                    USER AMAP..

80 00 00 28 18 72 db 5a    00 00 00 00 00 00 00 02    ...(.r.Z........
00 01 86 a0 00 00 00 02    00 00 00 04 00 00 00 00    ................
00 00 00 00 00 00 00 00    00 00 00 00                ............

00 0c 00 00 10 00 00 00    00 00 00 00 00 00          ..............
00 a6 00 00 01 00 00 00    01 34 01 2c 00 00 08 00    .........4.,....
7f ff 4f 98 00 00 00 01    00 84 00 22 00 00 00 00    ..O........."....
01 01 28 44 45 53 43 52    49 50 54 49 4f 4e 3d 28    ..(DESCRIPTION=(
43 4f 4e 4e 45 43 54 5f    44 41 54 41 3d 28 53 49    CONNECT_DATA=(SI
44 3d 74 65 73 74 29 28    43 49 44 3d 28 50 52 4f    D=test)(CID=(PRO
47 52 41 4d 3d 29 28 48    4f 53 54 3d 5f 5f 6a 64    GRAM=)(HOST=__jd
62 63 5f 5f 29 28 55 53    45 52 3d 29 29 29 28 41    bc__)(USER=)))(A
44 44 52 45 53 53 3d 28    50 52 4f 54 4f 43 4f 4c    DDRESS=(PROTOCOL
3d 74 63 70 29 28 48 4f    53 54 3d 31 36 32 2e 32    =tcp)(HOST=162.2
```

```
  37 2e 35 39 2e 31 33 35    29 28 50 4f 52 54 3d 31    7.59.135)(PORT=1
  35 32 31 29 29 29                                     521)))

  6c 00 0b 00 00 00 12 00    10 00 00 00 4d 49 54 2d    l...........MIT-
  4d 41 47 49 43 2d 43 4f    4f 4b 49 45 2d 31 00 00    MAGIC-COOKIE-1..
  c6 17 34 b7 89 ed 65 c0    93 fd d8 56 66 fa 52 40    ..4...e....Vf.R@

  12 01 00 34 00 00 00 00    00 00 15 00 06 01 00 1b    ...4............
  00 01 02 00 1c 00 0c 03    00 28 00 04 ff 08 00 00    .........(......
  c2 00 00 00 4d 53 53 51    4c 53 65 72 76 65 72 00    ....MSSQLServer.
  ac 07 00 00                                           ....
3c 20 4e 54 50 2f 31 2e    32 20 3e 0a                < NTP/1.2 >.
02 03 00 00 4b 00 00 00    00 00 00 00 00 02 65 6e    ....K.........en
  00 ff ff 00 1c 73 65 72    76 69 63 65 3a 74 68 63    .....service:thc
  3a 2f 2f 31 32 37 2e 30    2e 30 2e 31 3a 31 33 33    ://127.0.0.1:133
  37 00 00 0b 73 65 72 76    69 63 65 3a 74 68 63 00    7...service:thc.
  07 64 65 66 61 75 6c 74    00 00 00                   .default...
```

Az amap nem tételezi fel, hogy hogy a 22-es porton csakis és kizárólag ssh futhat, ezért különböző protokollok parancsait küldi el, mintha pl. Oracle-, MS SQL-, SMTP-, POP3 vagy NTP szerver lenne a túloldalon. Természetesen az ssh démon ezekre legtöbbször a ***„Protocol mismatch”*** üzenettel válaszol. Más kiszolgáló másfajta hibaüzenettel válaszolna, esetleg válasz helyett egyszerűen megszakítaná a kapcsolatot, azonban mindez segít, hogy kitaláljuk, mi van a másik oldalon. A hálózati forgalom alapján az is megállapítható, hogy minden portra ugyanazt az adathalmazt küldi el az amap, ami teljesen logikus, ha figyelembe vesszük, hogy előfeltételezések nélkül vizsgálja a célpont portjait.

Mindezek után hiába változtatja meg egy rendszergazda pl. az ssh portját 22-ről mondjuk 2222-re, az amap ki fogja találni, hogy ssh fut rajta. Így pl. hiába „rejtettem el” a mysql portomat 3307/tcp-re vagy 1234/tcp-re, az amap hibátlanul felismerte:

```
Protocol on 127.0.0.1:3307/tcp matches mysql
Dump of identified response from 127.0.0.1:3307/tcp (by trigger http):
0000:  3400 0000 0a35 2e30 2e36 3700 be87 0000    [ 4....5.0.67..... ]
0010:  296d 6a4e 2775 3b66 002c a208 0200 0000    [ )mjN'u;f.,...... ]
0020:  0000 0000 0000 0000 0000 006d 7273 6a3d    [ ...........mrsj= ]
0030:  3077 2a66 4667 5700 1000 0001 ff13 0442    [ 0w*fFgW........B ]
0040:  6164 2068 616e 6473 6861 6b65              [ ad handshake     ]
```

Ha már megvan, hogy milyen protokoll érhető el egy adott porton, akkor jó lenne azt is tudni, hogy milyen démon melyik verziója fut rajta - hátha egy ismert biztonsági hibát tartalmaz. Természetesen nem mindegyik szolgáltatás bannerje mondja meg, hogy „x program y verziója vagyok”, de a vmap segíthet ebben.

Töltsük le a http://freeworld.thc.org/releases.php?q=vmap&x=0&y=0 címről, és telepítsük az amap-nál leírt parancsokkal. A program jelenleg csak az **FTP, SMTP, POP3, IMAP és HTTP** protokollokat támogatja. Már itt szeretném megjegyezni, hogy egy régi – 2003-as – alkalmazásról van szó, és ennek megfelelően a felismert verziók is kissé porosak. A programnak opcionálisan egy login nevet és jelszót is meg lehet adni. Ha ezt megtesszük, akkor be is jelentkezik a megadott gépre. Ennek az az egyszerű oka, hogy sok parancs – amelyeket fel tud használni a verzió detektáláshoz - csak authentikáció után érhető el.
A vmap azt a jelenséget használja ki, hogy a különböző programok különböző verziói sajátos, egyedi hibaüzeneteket adnak a különféle protokollbeli hibákra. Ha ezeket -mint egy ujjlenyomatot- egy adatbázisban rögzítjük, akkor ezek alapján meghatározható a program verziója.

Tegyünk egy próbát, és adjuk ki az alábbi parancsot!

```
./vmap -l loginnev-p jelszo mail.aaaa.fu pop3
```

A program bejelentkezik a megadott adatokkal, majd az alábbi parancsokat küldi el a távoli POP3 kiszolgálónak:

```
62 6c 61 62 6c 61 0d 0a                                  blabla..
 53 54 41 54 0d 0a                                        STAT..
 53 54 41 54 20 62 6c 61    62 6c 61 0d 0a                STAT blabla..
 4c 49 53 54 0d 0a                                        LIST..
 4c 49 53 54 20 35 30 30    30 0d 0a                      LIST 5000..
 52 45 54 52 0d 0a                                        RETR..
 52 45 54 52 20 35 30 30    30 0d 0a                      RETR 5000..
 44 45 4c 45 0d 0a                                        DELE..
 44 45 4c 45 20 35 30 30    30 0d 0a                      DELE 5000..
 4e 4f 4f 50 0d 0a                                        NOOP..
 4e 4f 4f 50 20 62 6c 61    62 6c 61 0d 0a                NOOP blabla..
 52 53 45 54 0d 0a                                        RSET..
 52 53 45 54 20 35 30 30    30 0d 0a                      RSET 5000..
 54 4f 50 0d 0a                                           TOP..
 54 4f 50 20 35 30 30 30    0d 0a                         TOP 5000..
 54 4f 50 20 35 30 30 30    20 35 30 30 30 0d 0a          TOP 5000 5000..
 54 4f 50 20 35 30 30 30    20 2d 35 30 30 30 0d 0a       TOP 5000 -5000..
 55 49 44 4c 0d 0a                                        UIDL..
 55 49 44 4c 20 35 30 30    30 0d 0a                      UIDL 5000..
 55 49 44 4c 20 35 30 30    30 20 62 6c 61 62 6c 61       UIDL 5000 blabla
 55 53 45 52 0d 0a                                        USER..
 55 53 45 52 20 42 4c 41    42 4c 41 0d 0a                USER BLABLA..
 50 41 53 53 0d 0a                                        PASS..
 50 41 53 53 20 62 6c 61    62 6c 61 0d 0a                PASS blabla..
 41 50 4f 50 0d 0a                                        APOP..
 41 50 4f 50 20 62 6c 61    62 6c 61 20 63 34 63 39       APOP blabla c4c9
 33 33 34 62 61 63 35 36    30 65 63 63 39 37 39 65       334bac560ecc979e
 35 38 30 30 31 62 33 65    32 32 66 62 0d 0a             58001b3e22fb..
 41 50 4f 50 20 42 4c 41    42 4c 41 20 42 4c 41 42       APOP BLABLA BLAB
 4c 41 0d 0a                                              LA..
 51 55 49 54 0d 0a                                        QUIT..
```

Jól látható, hogy a program korrekt POP3 parancsok mellett egy halom érvénytelent is elküld. Sőt, az ezekre adott válasz még érdekesebb is, hiszen ezekre ***„-ERR valami_hiba_üzenet”*** válasz fog jönni, ami sok esetben az adott alkalmazásra jellemző. Világos, hogy ha a másik, a fejlesztők oldalát nézzük, akkor az a legjobb stratégia, ha kerüljük a túl bőbeszédű és a nagy tömegtől eltérő hibaüzeneteket, ill. protokollbeli viselkedést.

A távoli gépen egyébként a popa3d 1.0.x verziója fut, amit –mivel a program adatbázisa sem egy mai gyerek– nem ismert fel, viszont készített egy unknown_pop3d.fp nevű állományt. Ezt el lehet küldeni a THC fejlesztőinek, hogy beletegyék a következő verzióba, de most elég annyi, hogy a pop3/wl könyvtárba bemásoltam popa3d-1.0.x néven. A programot újra futtatva pedig – micsoda meglepetés! - a ***„Remote Daemon guess: popa3d-1.0.x with 100.00%.”*** üzenetet adta vissza.

Nézzünk most egy példát FTP szerver detektálásra. Ebben az esetben az alábbi parancsokat küldi el – többek között - a vmap:

```
62 6c 61 61 61 0d 0a                                  blaaa..
 53 49 54 45 20 45 58 45    43 20 7c 25 30 32 30 64    SITE EXEC |%020d
 7c 0d 0a                                              |..
 4d 4b 44 0d 0a                                        MKD..
```

```
43 57 44 0d 0a                                  CWD..
53 54 4f 52 0d 0a                               STOR..
43 44 55 50 0d 0a                               CDUP..
53 4d 4e 54 0d 0a                               SMNT..
50 4f 52 54 0d 0a                               PORT..
50 4f 52 54 20 68 31 0d    0a                   PORT h1..
50 4f 52 54 20 31 2c 32    2c 33 2c 34 2c 35 2c 36    PORT 1,2,3,4,5,6
50 4f 52 54 20 2d 31 2c    2d 32 2c 2d 33 2c 2d 34    PORT -1,-2,-3,-4
2c 2d 35 2c 2d 36 0d 0a                         ,-5,-6..
54 59 50 45 0d 0a                               TYPE..
54 59 50 45 20 41 0d 0a                         TYPE A..
54 59 50 45 20 3f 0d 0a                         TYPE ?..
54 59 50 45 20 5a 0d 0a                         TYPE Z..
54 59 50 45 20 49 0d 0a                         TYPE I..
53 54 52 55 0d 0a                               STRU..
53 49 54 45 20 48 45 4c    50 0d 0a             SITE HELP..
53 49 54 45 20 48 45 4c    50 20 52 45 54 52 0d 0a    SITE HELP RETR..
53 49 54 45 20 49 44 4c    45 0d 0a             SITE IDLE..
53 49 54 45 20 49 44 4c    45 20 2d 31 0d 0a    SITE IDLE -1..
53 49 54 45 20 43 48 4d    4f 44 0d 0a          SITE CHMOD..
53 49 54 45 20 45 58 45    43 0d 0a             SITE EXEC..
53 49 54 45 20 45 58 45    43 20 31 30 0d 0a    SITE EXEC 10..
45 50 52 54 20 7c 31 7c    32 2e 33 2e 34 2e 35 7c    EPRT |1|2.3.4.5|
36 7c 0d 0a                                     6|..
45 50 52 54 0d 0a                               EPRT..
```

Jól látható, hogy van közöttük érvényes ill. még több érvénytelen parancs. Sajnos ebben az esetben sem ismerte fel, hogy vsftpd 2.0.x-ről van szó – pedig az 1.1-es verzió szerepel az adatbázisában. Ezúttal is készített egy unknown_ftpd.fp nevű állományt, amit a ftp/wl könyvtárba bemásoltam vsftpd-2.0.x néven. A programot újra futtatva pedig természetesen a ***„Remote Daemon guess: vsftpd-2.0.x with 97.80%.”*** üzenetet adta vissza.

Tettem egy próbát a http szerver detektálással is, és az alábbi meglepő eredményt kaptam:

```
Banner says: Apache/1.3.36 (Unix)
Fingerprinting...
Remote Daemon guess: .Microsoft-ISS-6.0.swp with 83.33%.
```

Ezért újra futtattam a „`./vmap -c apache-1.3.36 www.aaaa.fu http`” parancsot. Ennek hatására létrejött a http/wo/apache-1.3.36 fájl, ami a neki megfelelő ujjlenyomatot tartalmazza. A parancsot újrafuttatva megint csak azt hitte, hogy IIS 6.0-val van dolga. Én pedig bánatomban úgy döntöttem, hogy itt abbahagyom az írást.

Mind az amap, mind a vmap jól használható az alkalmazások azonosítására. A vmap-nak elkel ugyan egy kis segítség, de ha valaki rászánja az energiát, hogy ujjlenyomatokat gyűjtsön, az egy nagyon jól használható információ birtokába kerülhet. Arra figyeljünk, hogy egyik sem biztosít az nmap-nál megtalálható ún. csali (decoy) szkennelést. Ennek az az oka, hogy míg az nmap nem foglalkozik a TCP kapcsolatok felépítésével, addig az amap/vmap működéséhez ez szükséges, és a TCP válaszokat is látnia kell.

Sütő János

A cikkhez tartozó fórum címe:
http://www.flosszine.org/mi_van_ott_amap_vmap

Főszerkesztő, alapító:
Horváth Örs Apor
(rendszermérnök - Budapest)

Szerzők:
Jankovich Oszkár
(újságíró - Budapest)
Kovács Zsolt
(informatikus - Debrecen)
Medve Zoltán
(linux rendszergazda - Szeged)
Pfeiffer Szilárd
(szoftvermérnök - Mosonmagyaróvár)
Sütő János
(rendszermérnök - Budapest)
Szőke József
(informatikus - Mikepércs)
Vomberg István
(szoftvermérnök - Budapest)

Közreműködők:
Fekete Róbert
(BalaBit)
Torma László
(Ubuntu tag)

Címlap, logó:
Makay József
(SKL Projekt)

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

Köszönet az FSF.hu Alapitványnak a tárhelyért!