Abíčko

Časopis serveru abclinuxu.cz

Září 2006

Vychází také na CD-ROM jako příloha časopisu

Editoriál

Vítejte u čtení časopisu Abíčko.

Abíčko vychází jako měsíční příloha serveru http://www.abclinuxu.cz a obsahuje výběr toho nejzajímavějšího obsahu, který zde byl v minulém měsíci publikován. Touto formou chceme předat čtenářům informace v snadno čitelné podobě vhodné i pro tisk.

Cílem serveru http://www.abclinuxu.cz je pomáhat všem uživatelům Linuxu, nezávisle na jejich zkušenostech, platformě či použité distribuci. Motorem, který nás pohání vpřed, je idea vzájemné pomoci a spolupráce. Proto i velkou část obsahu tvoří samotní uživatelé. Zapojit se může kdokoliv, tedy i vy.

Na http://www.abclinuxu.cz najdete rozsáhlou databázi návodů na zprovoznění hardwaru pod Linuxem, velice aktivní diskusní fórum, podrobné návody a tutoriály, recenze, archiv ovladačů, informace o linuxovém jádře (včetně populárních Jaderných novin) i rozcestník po ostatních linuxových serverech. Novinkou posledních měsíců, která našla brzy odezvu, jsou blogy neboli internetové deníčky. Každý registrovaný uživatel si jej může založit a psát si do něj poznámky nejen o Linuxu.

V neposlední řadě chceme upozornit také na výkladový slovník pojmů a vznikající elektronickou učebnici Linuxu, na níž se můžete podílet i vy!

Náměty na články zasílejte do konference našich autorů: *info@abclinuxu.cz*. Sponzoring Abíčka a jiné formy reklamy si objednávejte na adrese: *info@stickfish.cz*. Ostatní dotazy směřujte na adresu: *info@abclinuxu.cz*.

Server http://www.abclinuxu.cz provozuje firma Stickfish s.r.o., která poskytuje profesionální služby v oblasti Linuxu firmám i jednotlivcům. Zabývá se hlavně bezpečností, instalacemi Linuxu a konfigurací síťových služeb. Více na http://www.stickfish.cz.

Editor a sazba: Vlastimil Ott

Typografické konvence

Ve výpisech `zdrojových textů` mohou být použity znaky `\\`. Značí přechod na nový řádek, který ovšem *není* součástí samotného zdrojového textu, byl přidán editorem z důvodu lepšího vzhledu případně nemožnosti text formátovat bez jejich použití.

Obsah

PIM pro GNU/Linux – 1 (Kontact)

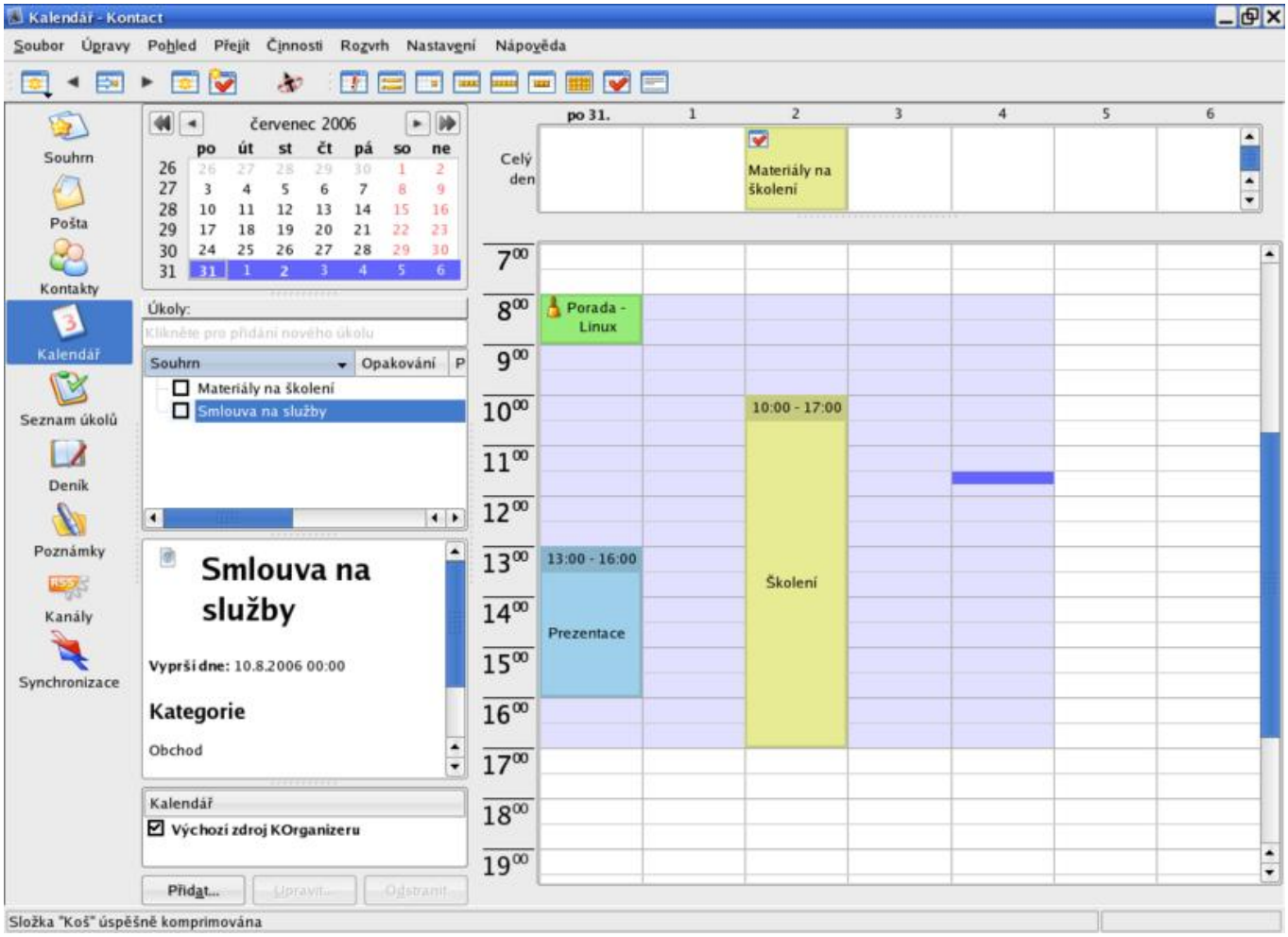
Lukáš Jelínek

Organizace času, komunikace s lidmi, psaní poznámek, podpora týmové práce – sloučením toho všeho získáme program, kterému se říká Personal Information Manager (správce osobních informací). V GNU/Linuxu jich máme v současné době na výběr hned několik.

Právě začíná krátký seriál, který by měl tyto programy představit a poskytnout každému zájemci informace k výběru toho správného. První díl bude věnován balíku Kontact [1] z desktopového prostředí KDE.

Kontact, KOrganizer a spol.

Systém Kontact vznikl integrací řady nástrojů, které již nějakou dobu existují. Lze je i nadále spouštět samostatně, ovšem spouštění v rámci balíku nabízí pohodlnější ovládání, lepší provázání a společnou konfiguraci. Kontact se v současné době skládá z těchto nástrojů (kromě uvedených aplikací Kontact využívá i některé doplňky):

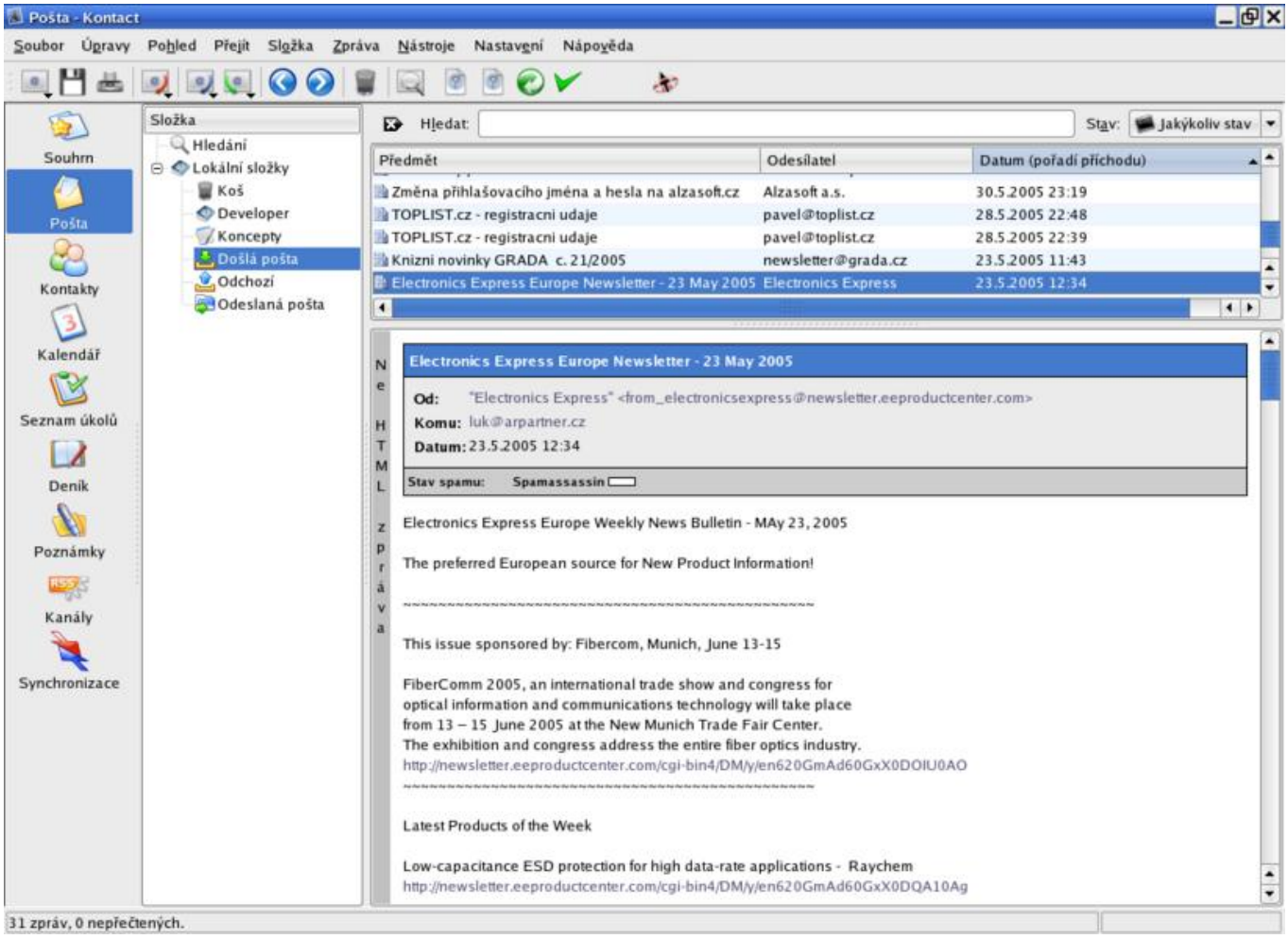


- *KOrganizer* – kalendář, plánování činností a událostí, správa úkolů, deník. Tato aplikace je velice propracovaná a funguje spolehlivě.
- *KMail* – poměrně známý poštovní klient, provázející prostředí KDE již od dávnověku. Opět velice propracovaný program, „našlapaný“ funkcemi. Funguje dobře, ale občas se špatně vyrovnává s některými typy příloh (při příjmu i odeslání).
- *KAddressBook* – z jednoduchého seznamu adres se postupem času stal mohutný správce kontaktů. Umožňuje pracovat s obrovským množstvím informací u každého kontaktu (včetně např. geografických údajů nebo nastavení šifrování zpráv), různě vyhledávat, filtrovat atd.

- *Akregator* – klient pro RSS kanály. Sice nenabízí příliš širokou škálu funkcí, zato se ale velice jednoduše ovládá. Je to dobré řešení pro ty, kdo nemají svou oblíbenou čtečku, ale přesto by rádi využívali technologii RSS.
- *MultiSynK* – umožňuje synchronizaci kalendáře, kontaktů a příp. i dalších dat mezi různými úložišti, místními i vzdálenými.
- *KPilot* – program pro synchronizaci dat mezi programem na PC (jak aplikací Kontact, tak i Evolution) a handheldem. Umožňuje též i pouhé zálohování dat z handheldu.
- *KNotes* – správce poznámek, které lze v podobě známých „žlutých papírků“ umísťovat na plochu. Tady je ovšem také slabina programu, protože přímo přes aplikaci Kontact to (zatím) provádět nelze – poznámka se tak pouze vytvoří, ale s jejím otevření na ploše je to složitější.
- *KNode* – klient pro diskusní skupiny (news). Uvádím ho jen pro úplnost, protože se v současných verzích Kontactu nezobrazuje. Lze ho zatím spouštět pouze samostatně.

Podpora groupware

Pro práci v týmu je prakticky nezbytné, aby aplikace PIM podporovala funkce pro pracovní skupiny – společné plánování času, adresáře a další věci. Kontact obsahuje podporu hned pro několik groupwarových serverů. Některé jsou podporovány plně (Kolab Server [2], SUSE Linux Openexchange [3], eGroupware [4], Novell GroupWise [5], Citadel [6]), jiné pouze částečně (OpenGroupware.org [7], Microsoft Exchange [8]). Množina je zcela dostatečná na to, aby si každý mohl vybrat to své řešení, a aby mohli spolupracovat i lidé, kteří mají různé klientské PIM programy.



Stav vývoje, použitelnost

Kontact v současné podobě je sice poměrně novou aplikací, ale přesto (hlavně vzhledem k jeho architektuře založené na samostatných komponentách) o něm lze říct, že je stabilní a vyspělý. Neobsahuje

podstatné chyby, které by uživatelům komplikovaly práci. Protože navíc jeho vývoj probíhal až dosud značnou rychlostí, pravděpodobně se ještě dočkáme dalšího zkvalitňování.

Výhody a nevýhody Kontactu

I když posouzení vlastností jako výhodné a nevýhodné je vždy subjektivní, lze nastínit aspoň nějaká základní vodítka, podle kterých se může orientovat zájemce o aplikaci PIM.

Výhody

- *Stabilita a vyspělost.* I když stále probíhá bouřlivý vývoj, Kontact lze již nějakou dobu prakticky bezproblémově používat. Případné chyby a nedostatky bývají rychle odstraňovány.
- *Funkční bohatost.* Oproti jiným podobným systémům obsahuje Kontact obrovské množství funkcí a nové rychle přibývají.
- *Založení na KDE.* Těsná vazba na KDE přináší jednak rychlý vývoj (díky využití množství hotových věcí), a samozřejmě i výbornou spolupráci s dalšími programy z tohoto desktopového prostředí (Konqueror apod.).
- *Dobrá groupwarová podpora.* I když hlavním groupwarovým serverem pro Kontact je z logického důvodu Kolab, lze pracovat i s jinými systémy. Lze navíc předpokládat, že se šíře podpory bude nadále zvětšovat.
- *Synchronizace.* Kontact již nyní poskytuje možnosti synchronizace s kapesními počítači i s úložišti dat. Navíc je již experimentálně použitelná aplikace *KitchenSync*, která brzy přispěje k dalšímu podstatnému zlepšení možností.
- *Nepříliš vysoké nároky.* Kontact nemá žádné přehnané nároky na rychlost procesoru ani na paměť. Dostatek paměti je jednoznačně přínosný, ale rozhodně nelze tvrdit, že by byl Kontact paměťovým žroutem.

Nevýhody

- *Možnost dezorientace.* Tím, že každá aplikace může fungovat jak samostatně, tak v rámci Kontactu (a často mírně odlišně), může uživatel ztratit orientaci v systému. Pokaždé si aplikaci spustí jinak a diví se, že se to chová jinak.
- *Založení na KDE.* Zdaleka ne každý využívá desktopové prostředí KDE. Proto se mu nebude líbit, že k funkčnosti Kontactu potřebuje mít nainstalovaný aspoň základ tohoto prostředí, tedy zejména knihovny Qt a KDE.
- *Drobné nepříjemné chyby.* Systém obsahuje různé drobné chybky (hlavně v GUI, ale i v back-endu), které sice nebrání práci, ale jsou občas otravné. Jako příklad lze uvést tuhnutí KMailu při některých operacích, nevhodné velikosti některých oken apod.
- *Malá přenositelnost.* V současnosti nejde Kontact používat jinde než na GNU/Linuxu. Jsou sice krkolomné cesty, jak to řešit, a dokonce se pracuje na implementaci KDE i pro jiné platformy, ale zatím je potřeba Kontact považovat za nepřenositelný.
- *Absence podpory silné firmy.* Toto je věc poměrně nepříjemná hlavně pro rutinní nasazení v prostředí větší firmy. Existují sice firmy schopné zajistit podporu pro provoz aplikací, ale těžko se lze dočkat rychlé reakce při nalezení nějaké vážnější chyby – a hlavně toho, aby se oprava takové chyby rychle dostala do hlavní větve (tím samozřejmě nechci nijak kritizovat vývojáře aplikací KDE, ale to provázání tu prostě není). Bude to zkrátka poněkud dobrodružná situace.

Závěr

Kontact je stabilní a dobře použitelný PIM systém pro GNU/Linux. Lze ho doporučit zejména těm, kdo rádi pracují s KDE a ocení tedy podobný vzhled aplikace a její integraci do prostředí. Nemusejí se ho bát

uživatelé se slabšími počítači, protože nemá příliš vysoké nároky na výkon. Naopak kvůli chybějící podpoře silné firmy se moc nehodí do velkých firem, kde není místo pro improvizace a partyzánská řešení. Příští díl bude věnován druhé z nejznámějších PIM aplikací, programu Evolution [9].

Odkazy

[1] http://kontact.kde.org/
[2] http://www.kolab.org/
[3] http://www.novell.com/documentation/suseslox/index.html
[4] http://www.egroupware.org/
[5] http://www.novell.com/products/groupwise/
[6] http://www.citadel.org/
[7] http://opengroupware.org/
[8] http://www.microsoft.com/cze/servers/exchange/
[9] http://www.gnome.org/projects/evolution/

⋆⋆⋆

PIM pro GNU/Linux – 2 (Evolution)

Lukáš Jelínek

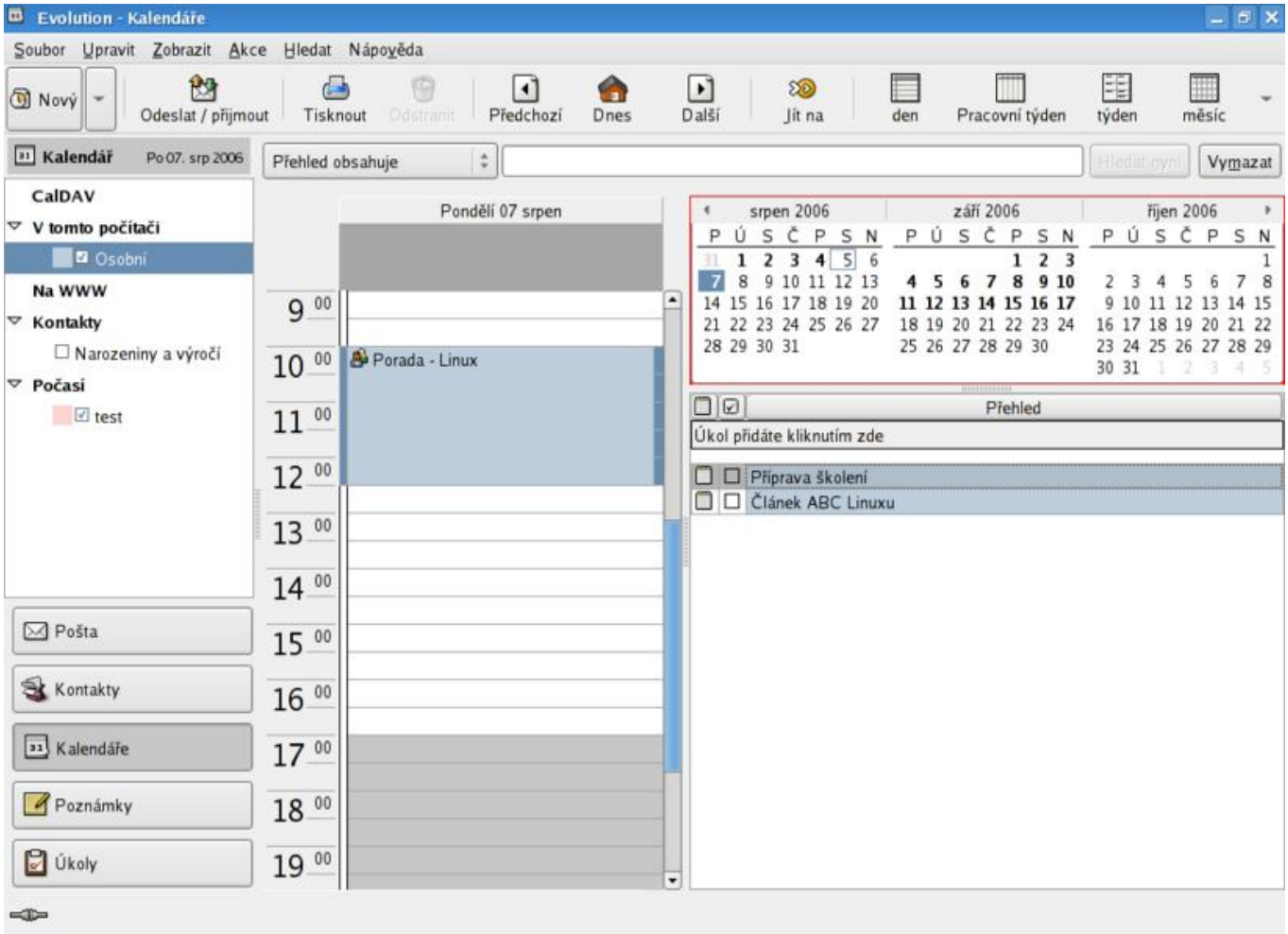
Program Evolution, velice známý zástupce aplikací PIM, je ze značné části pravým opakem minule představeného balíku Kontact. Je postaven na GTK+ verze 2, využívá knihovny desktopového prostředí GNOME, tvoří ho jediná velká aplikace a vývoj je podporován firmou Novell.

Podívejme se tedy, jakými vlastnostmi se aplikace Evolution vyznačuje, jak se s ní pracuje, a pro koho se hodí (popisovat budu verzi 2.6).

V jednoduchosti je krása

Po spuštění programu asi každého upoutá zjevná jednoduchost uživatelského rozhraní. Na rozdíl od Kontactu a dalších programů z KDE, které hýří barvami, je vzhled Evolution téměř fádní, což ale současně znamená rychlou orientaci v ovládacích prvcích, protože nikde nic nepřekáží a neupoutává pozornost.

Jak jsem již řekl v úvodu, jedná se o jediný program obsahující veškeré funkce pro různé oblasti PIM – tedy poštu, kontakty a plánování času. Nové funkce ale přidávat lze, a to pomocí zásuvných modulů (pluginů). S jejich pomocí se realizuje např. komunikace s groupwarovými servery, upozorňování na došlou poštu, automatické ukládání kontaktů a řada dalších věcí. Každý si samozřejmě může vytvořit vlastní plugin (třeba pro vazbu na firemní informační systém) a používat ho.



Přehled funkcí

I když aktuální soubor funkcí záleží na nainstalovaných a zapnutých pluginech, existuje základní společná množina, kterou najdeme všude:

- *pošta* – e-mailový klient s podporou řady komunikačních protokolů (včetně přístupu k datům uloženým lokálně). Kromě klasických složek používá tzv. složky hledání (dříve V-složky), což jsou pohledy na zprávy podle zvolených pravidel. I přes zdánlivou jednoduchost klient disponuje velkým množstvím funkcí.
- *kontakty* – lze kombinovat místní adresáře a servery LDAP. Množství ukládaných informací sice nedosahuje toho, co ukládá KAddressBook, ale pro normální použití je plně dostačující.
- *kalendáře* – Evolution pracuje hned s několika druhy. Jednak je to klasický místní kalendář, vzdálený (přístupný přes CalDAV nebo WebCal), kontaktový (narozeniny, svátky atd.) a poněkud nečekaně také „počasí“ (aktuální situace a předpověď pro další dny).
- *úkoly* – rozlišuje dva druhy úkolů: normální a přidělené. Pro normální úkoly lze nastavit obvyklé parametry (termín, stav atd.), kdežto přidělené používají mnohem složitější aparát, umožňující poměrně detailně určit, kdo nebo co se jakým způsobem na plnění úkolu podílí, včetně možnosti nechat si účastníky potvrdit účast.
- *poznámky* – v Evolution poměrně nová věc. Lze psát různé poznámky, třídit je do skupin a přidávat k nim přílohy.

Synchronizace a spolupráce s aplikacemi

Evolution využívá služeb programu gnome-pilot pro synchronizaci s PDA. Jiné synchronizace se musí řešit pomocí zásuvných modulů. V základní pluginové výbavě ale žádné takové bohužel nenajdeme. Jednou z vnějších aplikací pro synchronizaci je MultiSync [1] (nové verze budou vyvíjeny pod názvem OpenSync), poskytující mj. plugin pro Evolution.

O něco zajímavější je nyní možnost spolupráce Evolution s jinými aplikacemi, a to zejména programy gnome-panel (zobrazování událostí a úkolů na panelu), Planner (správa projektů; momentálně funguje pouze přenos kontaktů, pracuje se na těsnější spolupráci) a Gaim (instant messenger; přenos kontaktů z Evolution).

Podpora groupware

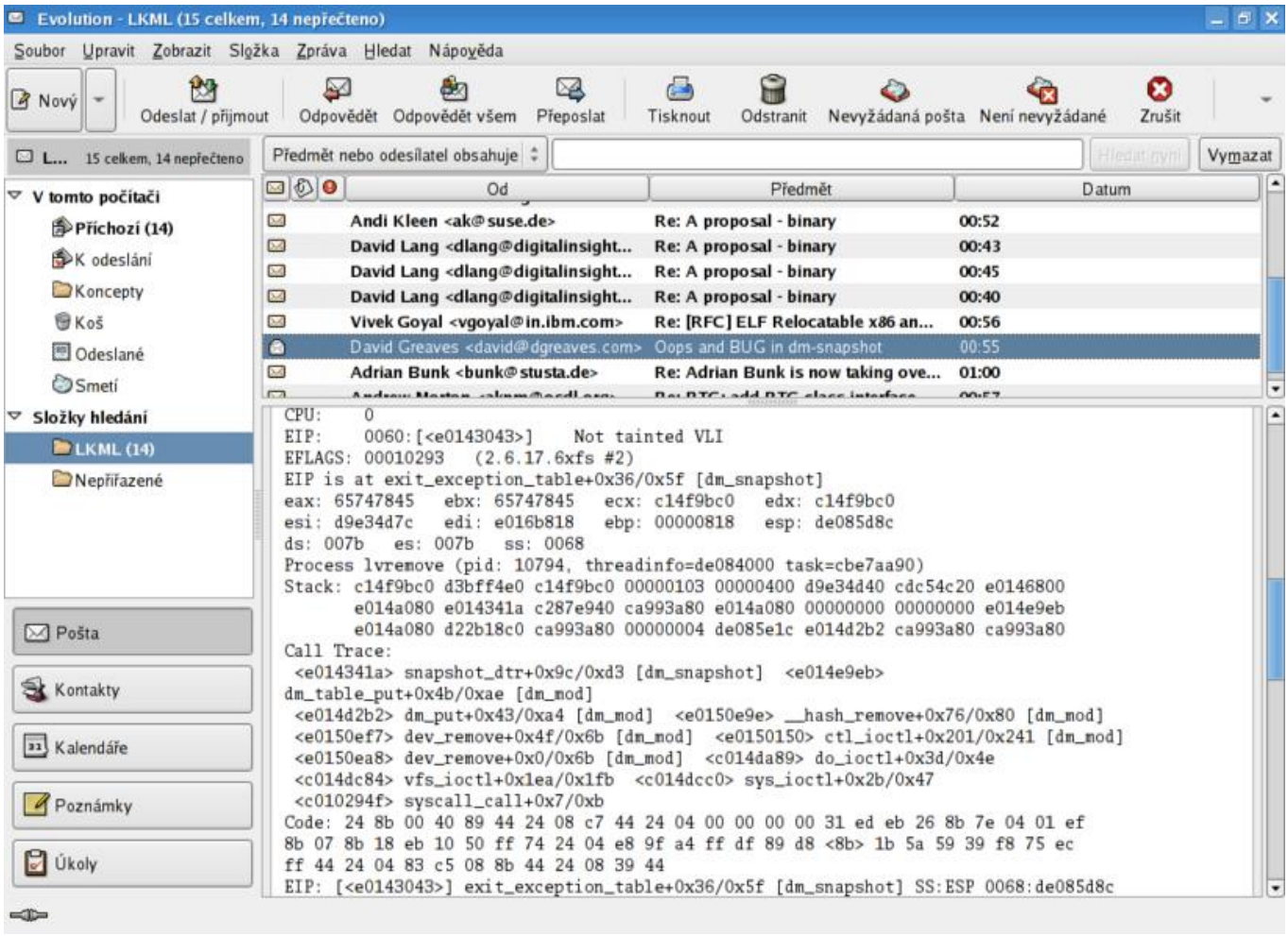
Groupware funkcionalita je řešena pomocí zásuvných modulů. V základní instalaci jsou podporovány servery Novell GroupWise [2] a Microsoft Exchange [3]. Nezávisle na hlavním projektu ale vznikly též pluginy pro další groupwarové systémy, např. OpenGroupware.org [4].

Stav vývoje, použitelnost

Program Evolution existuje již poměrně dlouho – nejprve ve firmě Ximian, která byla v roce 2003 koupena firmou Novell, a od té doby pak pod křídly tohoto známého síťařského giganta. I když se stále potýká s existencí drobných chyb (hlavně v pluginech), jedná se o vcelku vyzrálou a stabilní aplikaci, kterou je možno bez problémů nasadit i ve firemním prostředí v rámci přechodu na GNU/Linux.

Výhody a nevýhody Evolution

Podobně jako v případě Kontactu [5], i zde nyní představím hlavní výhody a nevýhody programu Evolution. Mohou posloužit při celkové orientaci v PIM aplikacích.



Výhody

- *Stabilita a vyspělost.* Evolution je programem, který své dětské nemoci prožil již dávno a netrpí tedy žádnými problémy čerstvých produktů. Vývoj probíhá poklidně a žádné živelné změny se nedějí.
- *Jednoduché ovládání.* Uživatelské rozhraní je navrženo velice jednoduše, což usnadňuje hlavně začátky práce s programem. Vzhled je střízlivý bez pouťových efektů, ke kterým trochu tíhne minule popisovaná aplikace Kontact.
- *Založení na GNOME.* Kdo využívá toto desktopové prostředí, uvítá shodný vzhled a ovládání GUI, a stejně tak i spolupráci s aplikacemi z rodiny GNOME. Další výhodou je, že právě GNOME bylo vybráno jako hlavní prostředí většiny enterprise distribucí, takže začlenění je o to lepší.
- *Dobrá podpora serveru Exchange.* V řadě firem, kde se na GNU/Linux přechází pouze částečně a stále se bude používat kombinace Exchange-Outlook, se velice hodí právě podpora zmíněného serveru. Může to být dokonce rozhodující kritérium, zda vůbec na Linux přejít.
- *Zásuvné moduly.* Každý si může poměrně snadno vytvořit zásuvný modul pro nějakou činnost a začlenit ho do Evolution. Je to opět výhoda pro heterogenní prostředí, kde se hodí každá možnost zlepšení interoperability.
- *Podpora silné firmy.* Hlavně větší firmy takovou podporu ocení. Mají k dispozici služby související s instalací, zaškolováním, řešením problémů atd.

Nevýhody

- *Méně funkcí.* Evolution nenabízí zdaleka tak širokou škálu funkcí jako Kontact. Takže kdo touží po nějaké méně obvyklé funkci, nebude pro něj Evolution dobrou volbou.

- *Založení na GNOME.* Podobně jako KDE, i GNOME má zaryté odpůrce, kteří nechtějí mít na svém stroji nějaké aplikace z tohoto prostředí. Dalším problémem jsou drobné chyby v knihovnách, které se často objevují s některými novými verzemi.
- *Chyby v distribucích.* Evolution v některých distribucích (momentálně např. FC 5) trpí určitými problémy. Nejčastěji se jedná o špatnou konfiguraci jak Evolution, tak dalších součástí distribuce (např. D-BUS).
- *Malá přenositelnost.* I když nedávno vznikla verze pro Windows, je to prakticky jen velice nestabilní technology preview (což bohužel platí i o knihovnách GTK+). Zatím platí, že Evolution lze používat pouze na GNU/Linuxu.
- *Slabá podpora synchronizace.* Jak jsem již říkal, samotný program žádnou podporu neobsahuje a musí spoléhat na pluginy a pomocné programy.
- *Hardwarové nároky.* Oproti Kontactu jsou nároky na procesor a paměť mírně vyšší, což ovšem vyplývá z použitého řešení GUI. Výkonnější počítač (řekněme jakýkoli novější než 2 roky) rozhodně není na škodu, i když na slabších sestavách se Evolution dá také přijatelně používat.

Závěr

Evolution je aplikace, která se hodí v situacích, kdy dáváme přednost stabilitě a jednoduchosti ovládání před množstvím funkcí. Typická oblast použití jsou firmy, kde se požaduje bezproblémové fungování, zejména při nutnosti kooperace s počítači, na nichž běží systém Windows. Lépe mu bude na silnějších počítačích, i když ani na těch slabších se není nutné Evolution vzdát. Při používání je potřeba dát pozor na kvalitu dané verze GTK+ a dalších používaných knihoven. V příštím dílu přijde na řadu méně známý, ale zato poměrně revoluční, program Chandler [6].

Odkazy

[1] http://multisync.sourceforge.net/
[2] http://www.novell.com/products/groupwise/
[3] http://www.microsoft.com/cze/servers/exchange/
[4] http://opengroupware.org/
[5] http://www.abclinuxu.cz/clanky/recenze/pim-pro-gnu-linux-1-kontact
[6] http://chandler.osafoundation.org/

⋆⋆⋆

PIM v GNU/Linuxu – 3 (Chandler)

Lukáš Jelínek

Představím vám program, který je sice poměrně málo známý, není ani ještě zdaleka dopracovaný, ale je současně velice revoluční. Přináší totiž do oblasti PIM řadu nových (nebo i starších, ale nerealizovaných myšlenek). Takže i kdyby sám o sobě příliš nezaujal, jeho koncepce může přinést mnoho pozitivního.

Nový pohled na věc

Chandler je desktopová PIM aplikace napsaná v Pythonu a poskytovaná s licencí GNU GPL. Pro uživatelské rozhraní využívá wxPython [1] a obsahuje vlastní framework pro vývoj modulů (zde zvaných *parcels*). Vývoj programu je od začátku veden novým pohledem na oblast PIM – vše by mělo být podřízeno tomu, jak s informacemi pracuje lidský mozek, nikoli vlastnostem technologií a komunikačních protokolů. Namísto různých druhů objektů (e-mailové zprávy, úkoly, události apod.) se zde rozlišují různé pohledy na objekty. Můžeme třeba naplánovat událost, poslat ji e-mailem (jen tak, nebo k ní připsat nějakou zprávu) a třeba z ní ještě udělat úkol. Tentýž objekt tak uvidíme v různých pohledech, jak z hlediska činnosti, tak ještě podle dalších kritérií (tzv. kolekce, např. odeslané/došlé, odstraněné, různé vlastní kolekce atd.). Nezáleží ani na tom, zda byl daný objekt vytvořen přímo v programu, přišel e-mailem, byl získán z kalendářového serveru, stažen pomocí RSS apod. Zatím není implementováno zdaleka všechno. To, co jsem uvedl výše, v tuto chvíli již funguje, i když něco působí na první pohled dost syrově. Na to, že má Chandler k první ostré verzi zatím dost daleko, je na tom překvapivě dobře. Marně ovšem nyní budeme v Chandleru hledat např. správu kontaktů, synchronizaci s přenosnými zařízeními, pokročilejší práci s poštou (přílohy, šifrování, filtrace…) a mnoho dalšího. Koho by zajímalo, ve které verzi by se měla ta či ona věc objevit, může si prohlédnout roadmap [2].

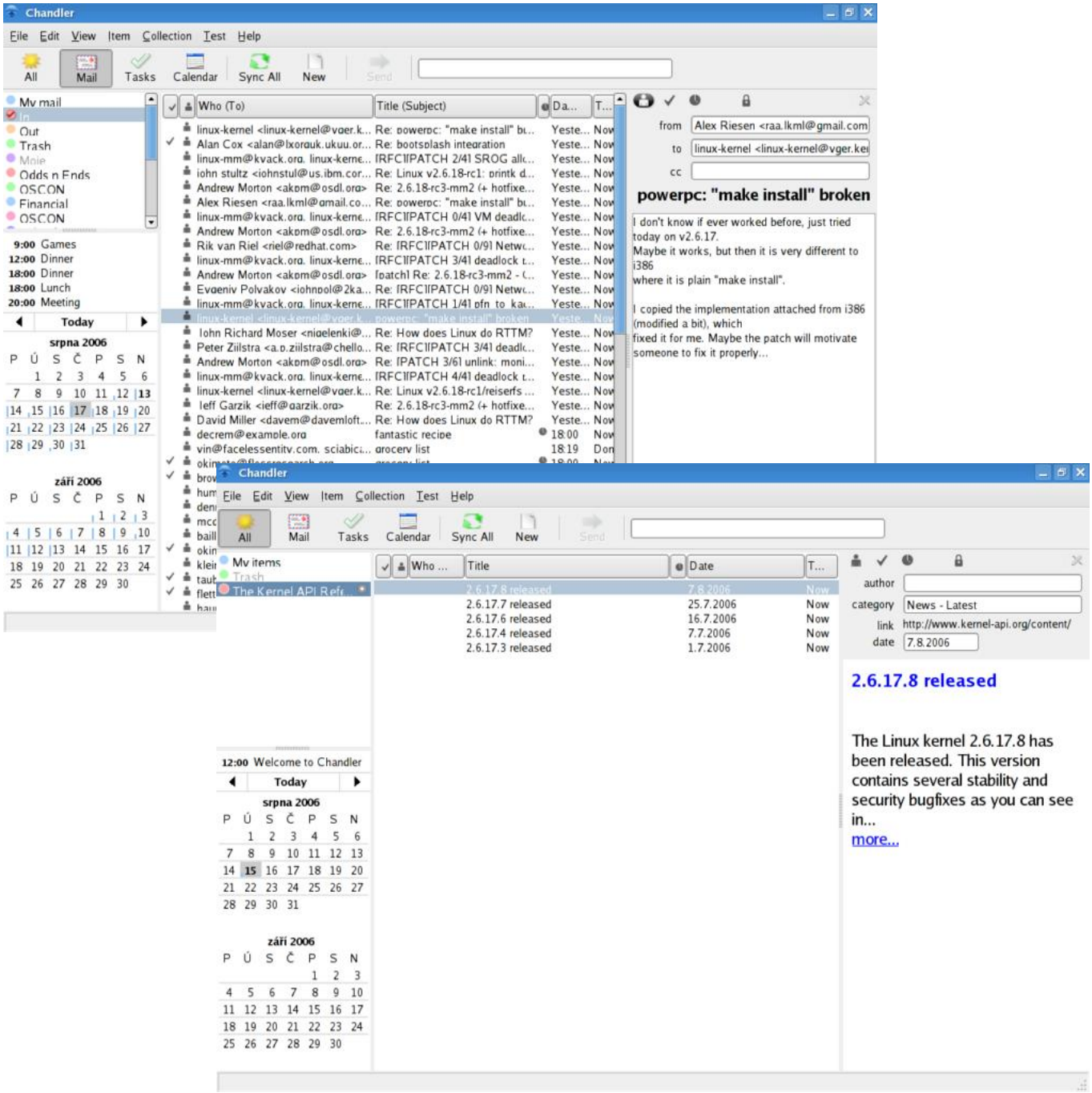
Výhody a nevýhody

Dobrých vlastností má Chandler na rozdávání. Podívejme se na ně blíže:

- *Multiplatformní řešení.* Chandler je od začátku navrhován a implementován jako plně přenositelná aplikace. Byl tomu přizpůsoben i výběr technologií, takže se dá říct, že co funguje na jedné platformě, funguje i na jiné. Momentálně jsou k dispozici verze po GNU/Linux, Windows a Mac OS X.
- *Podpora spolupráce.* Většina PIM systémů, které lze používat pod GNU/Linuxem, v tomto pokulhává. Na dostatečné úrovni je pouze Evolution [3] a v poslední době hlavně Kontact [4], ostatní programy se moc nevyznamenávají. Chandler je od počátku pro spolupráci navrhován, současně s ním je totiž vyvíjen i server Cosmo [5]. Ten je určený pro správu sdílených dat (použití serveru ovšem není omezeno jen na Chandler, je totiž založen na WebDAV/CalDAV a mohou s ním pracovat i jiné aplikace).
- *Modulární struktura.* Každá funkčnost je implementovaná jako samostatný modul (parcel). Díky tomu lze do Chandleru snadno přidávat další funkce, např. konverzní filtry pro různé datové formáty, nové funkce UI, funkce pro potřeby nějaké firmy apod. Autoři hovoří dokonce o „platformě Chandler“.
- *Pohledy na data.* Již jsem to zmínil, ale opět tuto věc zdůrazňuji. U e-mailu to už do jisté míry funguje třeba u M2 v Opeře nebo i „složek vyhledávání“ v Evolution [6], ale Chandler je na tomto postaven úplně celý.
- *Jednoduché grafické rozhraní.* Mnohé programy trpí přílišnou složitostí UI, ve kterém se nelze pořádně orientovat a práce s ním není efektivní. Chandler má GUI velice jednoduché (podobně jako třeba Evolution [7]), člověk se v něm neutopí a práce jde rychle od ruky.
- *Webové rozhraní.* I když samotný Chandler je běžnou desktopovou aplikací, existuje projekt Scooby [8] zabývající se vývojem webového rozhraní pro spolupráci s programem Chandler. Rozhraní je

určeno pro použití se serverem Cosmo (lze používat ovšem i s jinými CalDAV servery) a při jeho vývoji se uplatňuje stejná filosofie jako pro Chandler.

- *Zabudované testy ve vývojové verzi.* Vývojáři mají zjevně dobře na paměti, jak důležité je uživatelské testování. Proto do vývojových verzí zabudovali sadu různých testů a pomocných funkcí, které lze aktivovat přímo z menu programu a které umožňují snadno testovat program a odhalovat chyby.



Abych jen nechválil, Chandler má také některé nepříjemné vlastnosti. Některé do verze 1.0 téměř jistě zmizí, jiné bohužel přetrvají nebo se dokonce zhorší. Dejme se do toho:

- *Hardwarová náročnost.* Souvisí s využitím Pythonu, i když lze předpokládat, že díky pozdějšímu vyčištění kódu a optimalizacím se to poněkud zlepší. Na druhou stranu přibudou nové funkce, takže to nakonec může být úplně naopak. Každopádně nelze počítat s tím, že by se s Chandlerem dalo rozumně pracovat na slabších strojích (v podstatě cokoliv vyrobené před rokem 2003 bude nepoužitelné).

- *Filosofické pojetí.* Někomu nemusí vyhovovat to, že bude mít všechno na jednom místě. Podobné to bude i u pohledů na data – někdo má prostě rád klasické složky a nechce, aby mu stejná data figurovala na více místech.
- *Stabilita, spolehlivost.* Chandler bude monstrum. I když je modulární, je to prostě velká aplikace a je těžké vše perfektně odladit tak, aby to fungovalo bezchybně. Tím spíš, že pokud by se měl Chandler používat v enterprise prostředí, je perfektní stabilita naprosto klíčovou vlastností. I když vývojářům věřím a očekávám, že Chandler bude fungovat dobře (ostatně současné alfa verze 0.7alpha3 problémy se stabilitou nemá), je to prostě hodně velké sousto a nebudou to mít jednoduché (stačí se podívat na vizi projektu [9]).
- *Chybějící lokalizace.* Program zatím není lokalizován, což je v tomto stadiu docela pochopitelné. Ovšem pro toho, kdo není příliš kamarád s angličtinou, to může být dostatečný důvod, aby Chandler zatím ignoroval.

Závěr

Kdo tedy Chandler dosud neznal, má možnost se s tímto zajímavým projektem seznámit, do budoucna uvažovat o jeho používání, případně se již nyní podílet na jeho vývoji. Třeba tím, že si nainstaluje aktuální „stabilní“ (0.6) nebo alfa (0.7) verzi a bude program testovat (což není nic těžkého, viz výše). V každém případě tady máme aplikaci, kterou nelze nechat bez povšimnutí. Příště se podíváme na kombinaci Mozilla Thunderbird + Lightning (a na ještě dožívající spojení Mozilla Suite + Calendar extension), což je sice méně funkcemi nabité, ale přitom velmi stabilní a dobře fungující řešení.

Odkazy

[1] http://www.wxpython.org/
[2] http://wiki.osafoundation.org/pub/Projects/ChandlerHome/roadmap.html
[3] http://www.abclinuxu.cz/clanky/recenze/pim-pro-gnu-linux-2-evolution
[4] http://www.abclinuxu.cz/clanky/recenze/pim-pro-gnu-linux-1-kontact
[5] http://cosmo.osafoundation.org/
[6] http://www.abclinuxu.cz/clanky/recenze/pim-pro-gnu-linux-2-evolution
[7] http://www.abclinuxu.cz/clanky/recenze/pim-pro-gnu-linux-2-evolution
[8] http://scooby.osafoundation.org/
[9] http://chandler.osafoundation.org/1.0_vision.php

⋆⋆⋆

Traffic shaping (patchování a instalace)

Max Devaine

Traffic shaping slouží ke kontrole rychlosti přenosu dat. Lze pomocí něj určovat i prioritu pro různé přenosy, čímž můžeme např. docílit lepší odezvy u her, které hrajeme přes internet, i přesto, že náš soukmenovec stahuje ve vedlejším pokoji z p2p sítí. Celkově je toho mnohem více a my si o tom něco povíme.

Pár plků o použitých programech, modulech a patchích

Naším cílem je získat sadu funkčních programů opatchovaných vším možným pro co nejširší možnosti (i když třeba pro některé zbytečnými). Některými patchi se dostanu do části, která patří spíše firewallu, ale snad mi bude odpuštěno.

iptables

Základní stavební kámen ke kontrole síťových spojení apod. Nemá cenu to nějak rozvádět, kdo nezná, nemusí číst dál. Distribuční verze nebude bohužel stačit a budeme muset patchovat, co to dá.

iproute2

Sada programů mezi nimiž je mj. program `tc`, který je nezbytný pro shaping. Opět se nespokojíme s distribučním balíčkem a budeme muset patchovat a kompilovat.

patch-o-matic-ng (POM)

Sada patchů pro iptables, která se neobejde bez kernelu. Na verzích těchto patchů velice záleží. V jedné verzi najdeme to a v jiné zase ne. Čím novější verze, tím méně patchů. Patche postoupí a jsou zařazeny do jádra a iptables. Několik patchů jsem nenašel nikde, a proto je nutné použít starší verze POM. Nyní několik patchů z POM, které nás budou zajímat:

connlimit:

Umožňuje nastavit počet spojení, jak třeba na IP, tak i na jednotlivých portech. To je dobré třeba v kombinaci se stahováním ze sítě BitTorrent a starších AP, které nemohly překousnout ohromné množství spojení produkovaných sítí BitTorrent. Uplatnění lze samozřejmě najít i u jiných věcí.

geoip:

I když jsem zastáncem absolutní volnosti, tak tento patch zahrnu. Modul umožňuje pomocí externí databáze blokovat spojení přicházející z určitého státu (asi i města). Nechcete, aby se na váš server nějak pokoušeli připojit tučňáci ze severního pólu? Tohle je pro vás to správné řešení.

ipp2p:

Tento modul je schopen identifikovat pakety z p2p sítí. Tím pádem je můžeme filtrovat.

IPMARK:

Pomocí tohoto modulu můžeme označovat různé pakety, a díky tomu je pak můžeme jednoduše filtrovat. K jejich rozeznání nám poslouží buď zdrojové IP adresy, porty nebo jiná rozšíření typu ipp2p apod.

TARPIT:

Na TCP úrovni umožňuje zvýšit dobu odezvy na určitých portech díky nestandardnímu nastavení v TCP/IP protokolech, které jsou vyvolány útočníkovými akcemi typu scanování, automatizovaných útoků atd. V dnešní době už je ale tarpit rozeznatelný a nevím, na kolik procent je účinný.

nth:

Jednoduše řečeno umí sloučit internetové kapacity (Load Balancing). Řekněme, že máme třeba 3 přístupy na internet po 50 kB/s. Pomocí tohoto modulu je můžeme jakoby spojit a mít přístup s rychlostí stahování 150 kB/s.

time:

Umožňuje nám zacházet s pakety pomocí času/data. Pokud třeba chceme shapovat jinak v noci a jinak přes den atd.

u32:

Slouží k filtrování paketů. Myslím, že si je i umí poznačovat jako iptables pomocí IPMARK, ale není to moc šťastné řešení, jelikož jsou potom problémy se stabilitou (nebo bývaly?).

ROUTE:

Umožňuje měnit paketům výchozí bránu.

layer7 filtr

Tento filtr umožňuje rozeznávat pakety na aplikační úrovni. Ve finále je to něco podobného jako ipp2p. Patch aplikujeme na kernel a iptables.

esfq

Dokáže rovnoměrně rozdělovat traffic mezi uživatele (programy, třídy ...). Když chcete třeba 1Mb linku rovnoměrně rozdělit po 200 kb, tak esfq je to pravé. Není obsažen v iproute2 a musí se patchovat spolu s kernelem. Naproti tomu stávající `sfq` dokáže rovnoměrně rozdělovat traffic pouze mezi TCP spojeními.

imq

Je virtuální zařízení imq0/1, kam se přesměruje veškerý provoz (nebo provoz, který chceme shapovat) a tím pádem dostaneme zařízení se vstupem či výstupem a můžeme na něm cokoliv ovlivňovat (třeba download a upload). Patchujeme s ním kernel a iptables.

Co a kde stáhnout

Pro nenechavce. Důvodem, proč u `wget` hned ze začátku používám parametr `--continue`, je to, že když stahování přeruším, tak pokračuji bez jakéhokoliv dopisování, což se mi zdá mnohem rychlejší. Nejjednodušší je stahovat přímo do dir/usr/src. Jedním z důvodů je POM, ale případné symlinky můžete linkovat jinam, to je na vás.

```
cd /usr/src
wget -c ftp://ftp.cz.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.27.tar.bz2
```

Kernel řady 2.6.17 ještě nepoužívám. Donedávna byly problémy i s některými patchi pro řadu 2.6.16. Na patchování používat zásadně vanilla kernel [1]. Distribuční kernel by se vám nemuselo podařit buď opatchovat nebo zkompilovat..

```
wget -c http://www.netfilter.org/projects/iptables/files/iptables-1.3.5.tar.bz2
wget -c http://ftp.netfilter.org/pub/patch-o-matic-ng/snapshot/patch-o-matic-ng-20060626.tar.bz2
wget -c http://ftp.netfilter.org/pub/patch-o-matic-ng/snapshot/patch-o-matic-ng-20060511.tar.bz2
```

Verze 20060511 je poslední, která obsahuje patch geoip, takžeme budeme muset patchovat 2x. Myslím, že od verze 20060707 už není zahrnut v POM patch pro ipp2p, takže bych se vyhnul novějším verzím

POM. Osobně mám vyzkoušenou verzi 20060626. Mohli bychom také patchovat ipp2p ručně, ale proč si zbytečně přidělávat práci.

```
wget -c http://www.linuximq.net/patchs/linux-2.6.16-imq2.diff
wget -c http://www.linuximq.net/patchs/iptables-1.3.0-imq1.diff
wget -c http://fatooh.org/esfq-2.6/esfq-2.6.15.1.tar.gz
wget -c http://umn.dl.sourceforge.net/sourceforge/l7-filter/netfilter-layer7-v2.2.tar.gz
```

Když jsem dělal experimenty, tak byl odkaz z domovské stránky projektu pouze na verzi 2.1, někde jsem našel odkaz na 2.2, ve které už jde opatchovat kernel 2.6.16 a teď koukám, že už je k dispozici i verze 2.3 pro kernel 2.6.17. Rozhodnutí ponechávám na vás, ale vzhledem k označení si myslím, že asi nebude fungovat na starších jádrech.

```
wget -c http://developer.osdl.org/dev/iproute2/download/iproute2-2.6.16-060323.tar.gz
```

Doplňující balíčky, které nejsou předmětem dnešní debaty, ale patří k výše zmiňovaným filtrům:

```
wget -c http://puzzle.dl.sourceforge.net/sourceforge/l7-filter/l7-protocols-2006-05-21.tar.gz
wget -c http://people.netfilter.org/peejix/geoip/tools/csv2bin-20041103.tar.gz
```

Nakonec by bylo ještě dobré, kdybychom si všechno rozbalili:

```
tar xvfj linux-2.6.16.27.tar.bz2
tar xvfj iptables-1.3.5.tar.bz2
tar xvfz iproute2-2.6.16-060323.tar.gz
tar xvfj patch-o-matic-ng-20060626.tar.bz2
tar xvfj patch-o-matic-ng-20060511.tar.bz2
tar xvfz esfq-2.6.15.1.tar.gz
tar xvfz netfilter-layer7-v2.2.tar.gz
```

Na závěr ještě vytvořit symlinky:

```
ln -s /usr/src/linux-2.6.16.27 /usr/src/linux
ln -s /usr/src/iptables-1.3.5 /usr/src/iptables
```

Patchování

Nejdříve opatchujeme iptables a kernel pomocí imq:

```
cd iptables
patch -p1 < ../iptables-1.3.0-imq1.diff
cd /usr/src/linux
patch -p1 < ../linux-2.6.16-imq2.diff
```

Opatchujeme iptables a kernel pomocí filtru layer7:

```
patch -p1 < ../netfilter-layer7-v2.2/kernel-2.6.13-2.6.16-layer7-2.2.patch
cd /usr/src/iptables
patch -p1 < ../netfilter-layer7-v2.2/iptables-layer7-2.2.patch
```

Musíme ještě nastavit příslušná práva pro nová rozšíření v iptables:

```
chmod +x extensions/.layer7-test
chmod +x extensions/.IMQ-test
chmod +x extensions/.IMQ-test6
```

Dále opatchujeme iproute2 a kernel pomocí esfq:

```
cd /usr/src/iproute2-2.6.16-060323
patch -p1 < ../esfq-2.6.15.1/esfq-iproute2.patch
cd /usr/src/linux
patch -p1 < ../esfq-2.6.15.1/esfq-kernel.patch
```

A nakonec jsme si nechali POM. Jsou dva způsoby, jak použít patche z obou verzí POM. Buď chybějící patche dodáme do novějších POM, nebo budeme 2x patchovat, čemuž osobně dávám přednost. Nejdříve ovšem budeme muset (pokud nechcete ručně upravovat Makefile v kernelu) opatchovat samotné POM. První patch je pro geoip. Naleznete ho zde: patch-geoip.patch [2] a na internetu se nachází v konferenci na lists.netfilter.org [3]. Tento soubor si uložte do /usr/src/ a můžeme se hned pustit do opatchování geoip:

```
cd /usr/src/patch-o-matic-ng-20060511
patch -p1 < ../patch-geoip.patch
```

Dále si opatchujeme connlimit a IPMARK v novější verzi POM. Patch: patch-connlimit.patch [4] a na internetu se nachází na patchwork.netfilter.org [5]. Tento patch si uložíme do /usr/src/ a hned můžeme opatchovat POM:

```
cd /usr/src/patch-o-matic-ng-20060626
patch -p1 < ../patch-connlimit.patch
```

Tak a nyní můžeme s klidným svědomím opatchovat kernel a iptables:

```
/usr/src/patch-o-matic-ng-20060511
./runme geoip nth
```

Jelikož jsme si vytvořili symlinky, tak nemusíme pracně zadávat cestu ke kernelu a iptables a můžeme jen potvrdit nabízené cesty. Kdyby se objevila nějaká otázka, tak `y` a `enter` je správná odpověď. Nyní přejdeme k ostatním patchům v novějším POM:

```
cd /usr/src/patch-o-matic-ng-20060626
./runme time u32 connlimit ipp2p IPMARK ROUTE TARPIT
```

Postup je stejný jako v předchozím případě. Pokud byste se rozhodli zkusit i jiné patche, třeba pomocí ručního výběru příkazem `./runme extra`, tak vás musím upozornit, že se můžete dočkat chybových hlášeních typu (toto je např. z neopatchovaného geoip):

```
Do you want to apply this patch [N/y/t/f/a/r/b/w/q/?] y
unable to find ladd slot in src /tmp/pom-8915/net/ipv4/netfilter/Makefile \\
  (./patchlets/geoip/linux-2.6/./net/ipv4/netfilter/Makefile.ladd)
```

A otázka na akci vyskočí znovu. Řešením je zadat `f` jako `force`. Patch se aplikuje, jediný problém je, že se neaplikuje zápis do příslušného Makefile v kernelu, a tudíž ani po výběru položky se daný modul nezkompiluje. To lze vyřešit. Jak jinak, než ručním dopsáním Makefile. Najdete si Makefile patche, který se jmenuje `Makefile.ladd`. V našem případě je to v podadresářích /patchlests/geoip/linux-2.6/net/ipv4/netfilter/Makefile.ladd. Obsah tohoto souboru přepište do Makefile v kernelu v příslušných podsekcích. Takže nějak takto:

```
cat /usr/src/patch-o-matic-ng-20060511/patchlets/geoip/linux-2.6/net/ipv4/netfilter/\\
  Makefile.ladd >> /usr/src/linux/net/ipv4/netfilter/Makefile
```

Některé patche nemusíte zkoušet vůbec, buď jsou zbytečné, nebo s nimi jádro nezkompilujete. Např. tyto:

```
rpc
rtsp-conntrack
sip-conntrack-nat
quake3-conntrack-nat
```

Nastavení kernelu

Osobně se okolo iptables a shapování držím pravidla: Co jde, tak zkompilovat do modulu. Konečné rozhodnutí je na vás. Hlavní položky v kernelu pro shaping jsou tyto:

```
Networking  --->
 [*] Networking support
```

```
   Networking options  --->
 [*] Network packet filtering (replaces ipchains)  --->
```

V těchto podsekcích zaškrtáme, co se dá:

```
Core Netfilter Configuration  --->
IP: Netfilter Configuration  --->
IPv6: Netfilter Configuration (EXPERIMENTAL)  --->
```

V této sekci se nacházejí shapovací filtry:

```
Networking  --->
 [*] Networking support
   Networking options  --->
QoS and/or fair queueing  --->
```

Pozor na imq. Položka, která vytváří virtuální imq, se nachází zde:

```
Device Drivers  --->
 Network device support  --->
  <M> IMQ (intermediate queueing device) support
```

Tady ještě přikládám můj přeplácaný config [6].

Kompilace a instalace

Nejtriviálnější věc nakonec, to už tu snad nemusím ani uvádět, ale pro úplnost. Hlavně nezapomeňte odinstalovat distribuční verze programů iproute2 a iptables!

iproute2

Tady pozor. Když se vyskytne při kompilaci nějaká chyba, tak ji to jen vypíše a pokračuje dál. Pokud tedy nebudete sledovat, zda tam nemáte nějaký error, tak se nakonci dozvíte, že je všechno ok, ale pak vám půlka věcí nepůjde. Podle mých zkušeností byste v Debianu měli mít nainstalované mj. tyto balíčky: libdb2-dev, bison, flex. U jiných distribucí to bude asi obdobné.

```
cd /usr/src/iproute2-2.6.16-060323
make
make install
```

iptables

```
cd /usr/src/iptables
make
make install
```

kernel

```
...
make
make install
make modules_install
...
```

Závěr

Tak, teď byste měli být vybaveni pro trochu lepší shaping. Pokračování bude na téma shapovacích skriptů – celé výsledné řešení. Určitě jsem se v některých věcech seknul, tak budu jen a jen rád, když mě v komentářích opravíte, či doplníte apod.:-)

Odkazy

[1] http://www.abclinuxu.cz/slovnik/vanilla
[2] http://www.abclinuxu.cz/data/devaine/patch-geoip.patch
[3] http://lists.netfilter.org/pipermail/netfilter-devel/2006-May/024303.html
[4] http://www.abclinuxu.cz/data/devaine/patch-connlimit.patch
[5] http://patchwork.netfilter.org/netfilter-devel/patch.pl?id=3456
[6] http://www.abclinuxu.cz/data/devaine/config

⋆⋆⋆

VideoLAN Client – 1 (instalace a ovládání)

Jiří Poláček

VLC je výkonný přehrávač videa a hudby, přičemž multimediální obsah může být uložen v souboru, na DVD (přístupný s menu i bez), přijímán jako internetové vysílání či získán z kamery, televizní karty (analog i DVB) apod. Výstup přehrávaného zdroje přitom nemusí nutně putovat přímo na monitor a do reproduktorů, ale může být dále vysílán po síti, k čemuž je s oblibou využíván.

K dispozici je pro celou řadu platforem včetně Linuxu a ovládat se dá jak z příkazové řádky, tak via grafické rozhraní, telnet či webový prohlížeč. Webové stránky projektu sídlí na adrese www.videolan.org [1].

Instalace

Pro mnohé distribuce se nabízí instalační balíček [2], který ovšem nemusí být zkompilován se všemi uživatelem požadovanými funkcemi – například podpora karet pro příjem digitální televize je ve výchozím nastavení vypnutá – takže možná bude nutno přistoupit ke kompilaci ze zdrojových kódů.

V adresáři se zdrojovými kódy programu získáme příkazem `./configure --help` dlouhý seznam parametrů pro překlad programu, v kterém je vhodné zaměřit se především na vlastnosti ve výchozím nastavení vypnuté, zda nám nebudou chybět. Kromě výše zmíněné podpory DVB to mohou být moduly pro čtení DVD, „skinovatelné“ grafické rozhraní, přístup k digitálním kamerám, vstupní moduly Video4Linux, podpora dálkového ovládání, jiného zvukového systému, hudebního formátu Real Audio atd. Mnohé funkce samozřejmě závisí na externích knihovnách, vyplatí se proto dopředu si projít seznam požadovaných knihoven na http://www.videolan.org/vlc/download-sources.html [3] a ujistit se, že nám v systému nechybí hlavičkové soubory – v opačném případě si je doinstalovat. Osobně jsem VLC ve verzi 0.8.5, které budu dále popisovat, kompiloval s následujícími volbami:

```
./configure --enable-dvdread --enable-dvdnav --enable-dvb --enable-dvbpsi \\
--enable-real --enable-aa --enable-ncurses --enable-skins2 --enable-v4l \\
--enable-xosd
```

Po té již stačí obligátní `make && make install`.

Ovládací rozhraní

Program se pouští příkazem `vlc`, za kterým může následovat spousta parametrů určujících kde, co a jakým způsobem se má přehrát, například jednoduché `vlc video.mpg` spustí přehrávání souboru video.mpg. Při laborování budou určitě velmi užitečné parametry `help`, `longhelp` a `advanced` pro vyvolání (různě obsáhlé) nápovědy a `-v` pro výpis ladicích informací (se třemi stupni úrovně, společně s parametrem `color` budou chyby zřetelné na první pohled). Pokud naopak bude potřeba upovídanost VLC omezit na minimum, bude nás zajímat parametr `quiet` (jak je vidět u příkladů dále, dlouhé verze parametrů jsou v textu uváděny bez výchozích pomlček `--`). Další užitečné informace o programu jsou ve webové dokumentaci [4], wiki [5] a uživatelském fóru [6].

Po spuštění programu se VLC snaží inicializovat některé z dostupných *ovládacích rozhraní* (většinou umožňuje obvyklé interaktivní akce jako pozastavení přehrávané hudby, přesun na zvolenou kapitolu filmu apod.). Požadované rozhraní můžeme specifikovat jeho jménem uparametru `-I` (respektive `intf`); jednu instanci programu VLC může zároveň kontrolovat více rozhraní, k tomu pak slouží parametry `extraintf` a `control`. Příklady:

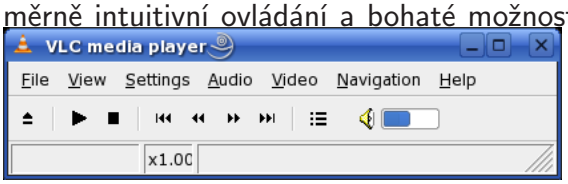
```
vlc -I ncurses video.mpg
```

```
vlc -I wxwidgets --control rc --extraintf http video.mpg
```

Nuže, co jsou jednotlivá rozhraní zač:

wxwidgets

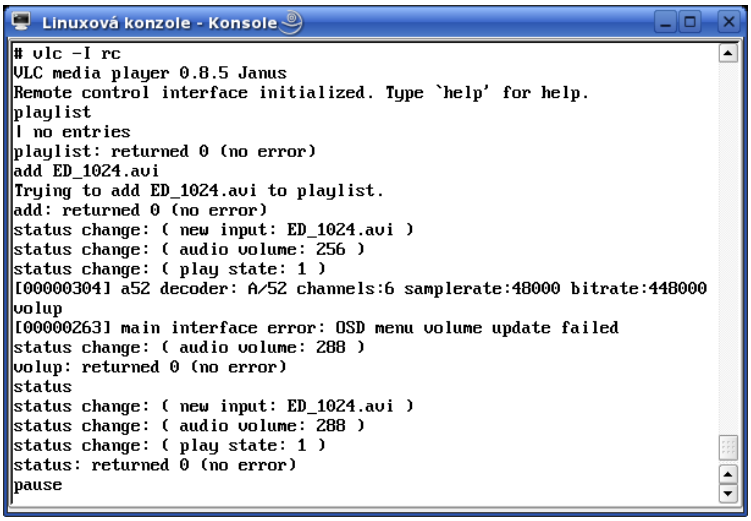
Základní grafické ovládací rozhraní, které se VLC snaží použít, pokud nebylo žádné jiné rozhraní specifikováno a zároveň VLC nebyl kompilován s parametrem `disable-wxwidgets`. Rozhraní má poměrně intuitivní ovládání a bohaté možnosti nastavení, zajímavým bonusem je průvodce (wizard) pro kódování/uložení vysílaného obsahu či naopak spuštění vlastního vysílání obsahu po síti. Pro spuštění VLC s tímto rozhraním se také instaluje alias `wxvlc`.



skins2

Grafické ovládání s možností změny vzhledu – několik „skinů“ lze stáhnout zdomovských stránek programu, krom nich však lze využít i soubory pro Winamp2 a XMMS. Toto rozhraní je v současné době označováno jako experimentální; přístupné je za předpokladu kompilace programu s parametrem `enable-skins2` a existuje pro něj alias `svlc`.

rc



Textová konzole, VLC se ovládá psaním příkazů jako `pause`, `forward`, `volume`, `playlist` apod., seznam dostupných příkazů se vypíše po zadání `help`. K tomuto rozhraní se dá připojit i vzdáleně skrze TCP/IP nebo unixový socket; rozhraní se použije automaticky v případě, že byl VLC kompilován bez podpory grafického rozhraní.

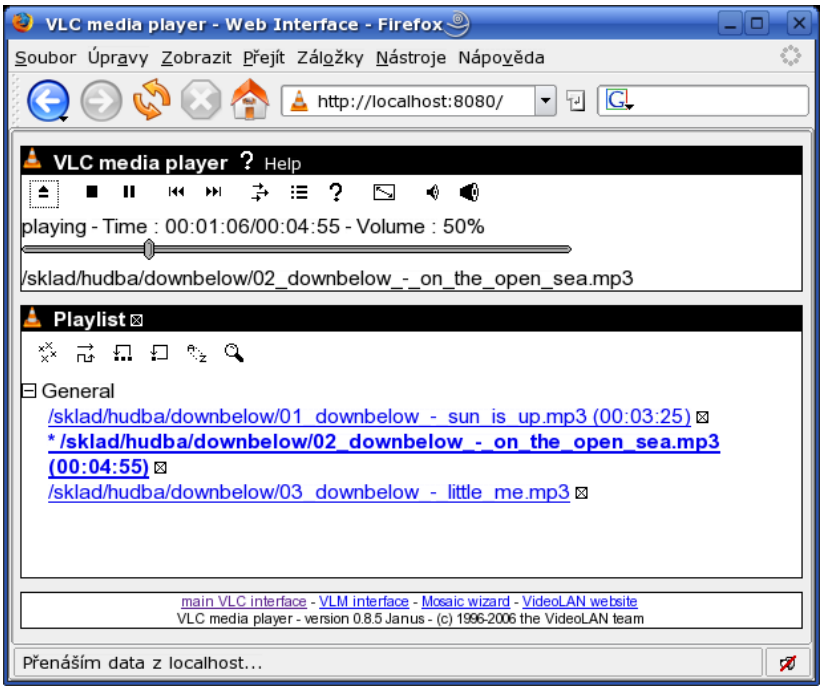
ncurses

Prostředí v textovém režimu, kde si uživatel vystačí s klávesovými zkratkami. Výchozí adresář po spuštění lze zadat s pomocí parametru `browse-dir`. Rozhraní je dostupné pouze, pokud byl VLC kompilován s parametrem `enable-ncurses`.

telnet

Vzdálené ovládání VLC prostřednictvím standardního telnetu. Bez upřesnění dalších parametrů server naslouchá na všech síťových rozhraních na portu 4212 a vyžaduje heslo *admin*; tyto údaje nastavují parametry `telnet-host`, `telnet-port` a `telnet-password`. Po přihlášení získáme výpis dostupných příkazů odesláním slova `help`.

http



Vzdálené ovládání VLC prostřednictvím webového prohlížeče s podporou JS. Síťové rozhraní a port se nastaví parametrem `http-host`, ve výchozím nastavení server naslouchá na všech síťových rozhraních na portu 8080. Rozhraní je dostupné, pokud při kompilaci VLC nebylo zakázáno parametrem `disable-httpd`.

gestures

Myší gesta, tj. předepsané pohyby kurzorem; použitelné pro základní ovládání přehrávače typu „skoč na další položku“ apod.

dummy

Triviální rozhraní, které činí VLC naprosto neovladatelným ;-) Hodí se v situacích, kdy není zapotřebí do přehrávání multimédií nikterak zasahovat, například při streamování televizního vysílání.

Klávesové zkratky

K ovládacím rozhraním se také řadí klávesové zkratky. Jsou vždy přístupné v okně videa, můžete vyzkoušet například mezerník pro pozastavení videa, klávesu `F` pro fullscreen, `Ctrl-Up` a `Ctrl-Down` pro ovládání hlasitosti či `Ctrl-Q` pro ukončení přehrávače. Pro spouštěnou instanci VLC lze klávesové zkratky definovat odpovídajícím parametrem na příkazovém řádku, například:

```
vlc video.mpg --key-vol-mute 'Ctrl-Shift-M'
```

Trvale lze (nejenom) vlastní klávesové zkratky nastavit v konfiguračním souboru – najdete jej v `~/.vlc/vlcrc`, pokud takový soubor zatím neexistuje, příkazem `vlc --save-config` jej vytvoříte. Popis skladby klávesových zkratek je dostupný ve webové dokumentaci [7], použitelné parametry pak v „dlouhé“ nápovědě – `vlc --longhelp`.

Odkazy

[1] http://www.videolan.org/
[2] http://www.videolan.org/vlc/
[3] http://www.videolan.org/vlc/download-sources.html
[4] http://www.videolan.org/doc/
[5] http://wiki.videolan.org
[6] http://forum.videolan.org
[7] http://www.videolan.org/doc/play-howto/en/ch04.html#id293710

⋆ ⋆ ⋆

VLC – 2 (přehrávání multimédií)

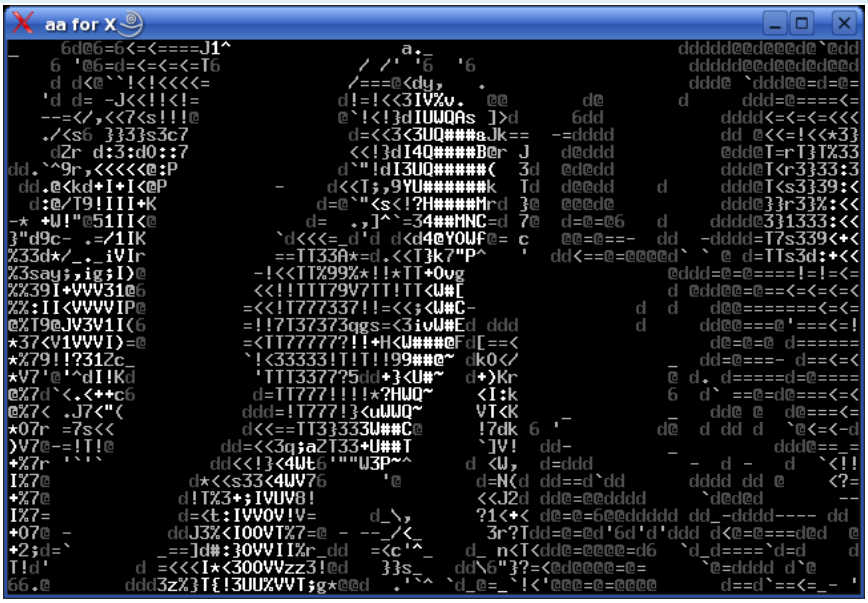
Jiří Poláček

Navzdory bohatému množství způsobů ovládání přehrávače VLC media player, kterým se věnoval první díl seriálu, se přidržíme při zemi a ukážeme si, jak z příkazové řádky pustit hudební céčéčko, filmové DVD a naladit internetové rádio či televizní vysílání.

Výstupní moduly zvuku a videa

Ještě než se dostaneme k odpovědi na otázku, co a jak přehrávat, zmiňme krátce *s pomocí čeho přehrávat* – u VLC jsou myslitelné prakticky všechny zvukové systémy (OSS, ALSA, ESD, aRts) i videovýstupy (X11, OpenGL, SVGAlib, framebuffer apod.) – samozřejmě za předpokladu, že byl VLC kompilován s podporou dotčené knihovny. Seznam všech dostupných modulů přehrávač vypíše po zadání `vlc -l`, vybraný modul specifikujeme pomocí parametru `aout` pro zvuk a `vout` pro video. Příznivci Ascii Art a OSS mohou zadat například:

```
vlc video.avi --aout oss --vout aa
```



Jiný netradiční příklad – pro získání série obrázků z videa:

```
vlc video.mpg --vout image --image-out-format jpg \\
 --image-out-prefix obrazek --image-out-ratio 30
```

Z každého třicátého snímku videa se v aktuálním adresáři vytvoří obrázek ve formátu JPEG (vybrat lze i PNG) se jménem složeným ze zadané předpony `obrazek` a pořadového čísla.

Playlisty a síťové zdroje

Již jsme zmínili, že pro přehrání nějaké hudby či videa stačí uvést cestu k příslušnému souboru jako parametr programu VLC. Dá se říci, že to obecně platí i pro seznamy nahrávek a síťové vysílání. Podporovány jsou playlisty ve formátech `M3U` a `PLS`:

```
vlc downbelow.pls
VLC media player 0.8.5 Janus
Remote control interface initialized. Type ‘help’ for help.
playlist
```

```
|*sun_is_up   01_downbelow_-_sun_is_up.mp3|downbelow.pls|
| on_the_open_sea   02_downbelow_-_on_the_open_sea.mp3|downbelow.pls|
| little_me   03_downbelow_-_little_me.mp3|downbelow.pls|
playlist: returned 0 (no error)
```

V konzolovém ovládacím rozhraní vypíše příkaz `playlist` aktuální seznam skladeb, příkaz `add` přidá další položku a s pomocí `next` a `prev` v seznamu skladeb vybíráme příští, respektive předchozí položku. Prázdný playlist nám zajistí příkaz `clear`. Ukázka naladění internetového rádia:

```
vlc http://www.live.cz/radio/beat128.ogg.m3u
VLC media player 0.8.5 Janus
Remote control interface initialized. Type 'help' for help.
status change: ( new input: http://www.live.cz/radio/beat128.ogg.m3u )
```

Při přehrávání internetového zdroje VLC nezastaví ani drobné překážky – pokud je třeba přistupovat přes proxy, lze ji definovat parametrem `http-proxy`; při přehrávání z FTP-serveru se lze autentizovat dvojicí `ftp-user` a `ftp-pwd` (v otevřené podobě). Multimediálních zdrojů lze uvést i více, přehrají se právě v tom pořadí, v jakém je uvedeme:

```
vlc vecernicek.avi vlc:pause:8 famfara.mp3 vlc:quit
```

Direktivy `vlc:pause` a `vlc:quit` mají v seznamu speciální účel – z názvu je patrné, že `vlc:pause` pozastaví další přehrávání po stanovenou dobu v sekundách a `vlc:quit` ukončí aplikaci (hodí se obzvláště u rozhraní `dummy` pro korektní ukončení VLC po skončení přehrávání).

Přehrávání ze speciálních zdrojů obsahu

Speciálními zdroji obsahu jsou zde míněny hudební cédéčka, filmová cédéčka a dévédéčka a karty pro příjem televizního vysílání. Typ zdroje specifikujeme podobně jako síťový protokol, případné vlastnosti zdroje pak s pomocí k tomu určených parametrů. Přehrajme si hudební CD:

```
vlc cdda://
```

Takto se VLC pokusí přehrát CD od začátku v zařízení, které vyčte z konfiguračního souboru či zvýchozího zařízení, kterým je `/dev/cdrom`. Můžeme upřesňovat:

```
vlc cdda:// --cd-audio /dev/dvdrecorder --cdda-track 9 --cdda-caching 400
```

Parametrem `cd-audio` vybereme mechaniku, která má cédéčko přehrát, `cdda-track` způsobí přehrání výhradně vybrané stopy a s pomocí `cdda-caching` upravíme velikost vyrovnávací paměti v milisekundách. VLC umí také spolupracovat se servery CDDB, k nastavení spojení slouží parametry `cddb-server` a `cddb-port`. Přehrávání filmového cédéčka je podobné:

```
vlc vcd:// --vcd /dev/cdrw --vcd-caching 250
```

Širší možnosti poskytuje přehrávání filmů na nosičích DVD. Pokud bylo VLC kompilováno s podporou `dvdnav`, lze využít přístup `dvd://` k plnohodnotnému prohlížení disku včetně menu; přístup `dvdsimple://` přímo spustí přehrávání filmu. Pokud chceme přehrát pouze určitý titul či kapitolu, máme možnosti podle následující šablony:

```
vlc dvd[simple]://[@[title][:[chapter][:angle]]]
```

Tedy například

```
vlc dvd://@1:5 --sub-language 'cs'
```

přehraje pátou kapitolu z prvního titulu, přičemž menu bude přístupné pro pozdější vyvolání. Příklad zároveň ukazuje výběr titulků podle kódu jazyka, alternativně lze titulky vybrat též pořadovým číslem

udaným za parametrem `sub-track`. K přehrávání DVD dodejme ještě, že správné zařízení, kde se nachází filmové DVD, upřesníme parametrem `dvd`.

Ladíme programy televizního vysílání

Věnujme se nejdříve tomu zajímavějšímu zdroji, kvůli kterému si mnohý cestu k VLC nachází – digitální televizi. Aby bylo možné ji sledovat a následně streamovat, musí být VLC kompilováno s podporou DVB. Aplikace bohužel nespolupracuje skonfiguračním souborem channels.conf, který je výsledkem známých ladících utilit `(t,s,c)zap`, potřebné údaje o vysílaní je třeba mít v konfiguračním souboru VLC (/.vlc/vlcrc) nebo zadat spomocí parametrů:

```
vlc dvb: --dvb-frequency=626000000 --dvb-bandwidth=8
```

Příklad se týká pozemního digitálního vysílání, přičemž dalších pět souvisejících parametrů má výchozí hodnoty shodné s těmi, s jakými se v naší zemi vysílá a tudíž není potřeba je explicitně zadávat. Výpis všech parametrů z nápovědy týkajících se DVB (včetně satelitní a kabelové verze) získáme příkazem `vlc -p dvb --advanced`.

Pokud neurčíme jinak, VLC z celého naladěného multiplexu vezme první vysílaný proud, na který narazí, a spustí jeho přehrávání; předchozí příklad tak konkrétně při signálu multiplexu A vysílaného zbrněnských Hádů pustí ČT4 Sport. Pro sledování jiného televizního kanálu budou zapotřebí další parametry:

```
vlc dvb: --dvb-frequency=626000000 --dvb-bandwidth=8 --ts-es-id-pid --program 1
```

Parametr `ts-es-id-pid` umožní odvolávat se na jednotlivé proudy číslem kanálu (lze vyčíst z channels.conf) a konečně parametrem `program` daný proud vybereme (v tomto případě ČT1). V případě analogového televizního vysílání spoléhá přehrávač na architekturu *Video4Linux* – za předpokladu, že byl zkompilován s parametrem `enable-v4l`. V takovém případě je samozřejmě myslitelný přístup i k jiným podporovaným zařízením, jako jsou webové kamerky apod. Bohužel nemohu ověřit, takže pouze ocituji příklad přístupu k podobnému zařízení:

```
vlc v4l:// --v4l-vdev=/dev/video --v4l-adev=/dev/dsp
```

Přístup k architektuře Video4Linux udává `v4l://`, následující dva parametry specifikují zařízení, odkud se má brát video a zvuk. Veškeré přípustné parametry vypíšeme `vlc -p v4l --advanced`.

Příště

Následující díl seriálu o VLC se bude zabývat pokročilejšími aspekty přehrávání multimédií, řeč bude bude zejména o nasazení titulků a obrazových filtrů.

⋆ ⋆ ⋆

VLC – 3 (filtry a titulky)

Jiří Poláček

Pro vymezení vlastností přehrávání disponuje VLC celou řadou nastavení a filtrů – představíme si nejpoužívanější a nejzajímavější z nich. Většina se samozřejmě dá interaktivně aplikovat zovládacího rozhraní přehrávače, zde se však opět přidržíme příkazové řádky.

Základní vlastnosti přehrávání

Opakované a náhodné přehrávání mají na starosti parametry `repeat`, `input-repeat`, `loop`, `random`:

```
vlc --loop --random --input-repeat 1 video1.avi video2.avi video3.avi
```

Přehrávač bude tři uvedená videa přehrávat náhodně v nekonečné smyčce, přičemž každé z nich dvakrát po sobě (hodnota parametru `input-repeat` přidává udaný počet přehrání k výchozímu jednomu přehrání, proto zde $1 + 1 = 2$).

```
vlc --repeat video1.avi video2.avi video3.avi
```

Přehrávač bude každé video přehrávat opakovaně, dokud jiným způsobem (například příkazem `next` v ovládací konzoli) nepřejdeme na další položku seznamu skladeb. Jakýkoliv zdroj s náhodným přístupem (tj. například písničku uloženou na disku, nikoliv však již televizní vysílání) není nutno přehrávat od začátku do konce. Parametry `start-time` a `stop-time` vymezují, od kolikáté do kolikáté sekundy se má přehrávat:

```
vlc trailer.mov --start-time 60 --stop-time 120
```

Z filmové ukázky se přehraje pouze druhá minuta.

Velikost na přání

Ve výchozím nastavení se video začne přehrávat v okně, jehož velikost se přizpůsobí velikosti tohoto videa. Velikost okna lze nicméně nastavit ručně – napevno zadáním šířky a výšky (parametry `width` a `height`) nebo poměrově parametrem `zoom`. Nechybí ani možnost přehrávání přes celou obrazovku, příslušný parametr se jmenuje `fullscreen`.

```
vlc --width 800 --height 600 video.avi
vlc --zoom 0.5 video.avi
vlc --fullscreen video.avi
```

V příkladech bude video postupně přehráno v okně o velikosti 800×600 bodů, s poloviční velikostí a přes celou obrazovku. Splnitelné je i přání žádné obrazové složky u videa s pomocí parametru `novideo` – chceme si pouze poslechnout audio. Analogicky `noaudio` vypíná zvuk; hlasitost lze jinak upravit zadáním hodnoty z intervalu 0–1024 za parametr `volume`:

```
vlc --novideo --volume 512 video.avi
```

VLC z videa přehraje pouze zesílený zvuk (výchozí úroveň hlasitosti je na hodnotě 256).

Titulky v přehrávači

Podpora titulků je ve VLC na velmi dobré úrovni, přehrávač si rozumí s širokou plejádou nejrůznějších formátů, problémem nejsou různá kódování, umístění titulků či barva, průhlednost a velikost písma. Z ladícího výpisu lze také vyčíst, že program sám hledá titulky v adresáři s filmem, rozpoznává jejich typ a automaticky je načítá – pihou na kráse je fakt, že v testované verzi 0.8.5 se přesto titulky nezobrazí

(v nadcházející verzi již by mělo být opraveno). Chceme-li tedy zobrazit titulky, musíme zadat cestu k souboru parametrem `sub-file`. Dalším zádrhelem může být výchozí font FreeSerifBold.ttf, který se VLC snaží použít – pokud v systému chybí, je třeba zadat alternativu:

```
vlc video.avi --subsdec-encoding CP1250 --sub-file video.sub \\
--freetype-font /usr/X11/lib/X11/fonts/truetype/arial.ttf
```

Parametr `subsdec-encoding` specifikuje znakovou sadu titulků (pokud neodpovídá skutečnosti, VLC titulky vůbec nezobrazí), pro nás budou zejména užitečné hodnoty `CP1250`, `ISO-8859-2` a `UTF-8`. Nyní si zkusme pohrát s umístěním:

```
vlc video.avi --sub-margin 30 --subsdec-align 1 --sub-delay -50
```

Hodnota `1` u `subsdec-align` znamená zarovnání titulků vlevo, výchozí `0` značí obvyklé centrování textu a `2` pak zarovnání vpravo. Hodnota u `sub-margin` definuje velikost mezery v obrazových bodech od spodního okraje okna k titulkům, v příkladě tedy budou titulky posunuty o 30 bodů nahoru. O relativní umístění v čase se pak stará `sub-delay`, jednotkou jsou zde myšleny snímky, `-50` tedy způsobí, že titulky se budou zobrazovat o nějaké dvě sekundy dříve než určuje jejich definice. A jak na vzhled titulků? O fonty se stará knihovna `Freetype2`:

```
vlc video.avi --freetype-color 16776960 --freetype-effect 3 \\
--freetype-opacity 128 --freetype-rel-fontsize 12
```

Kód barvy 16776960 opsaný z nápovědy značí žlutou barvu, `freetype-effect` přidává písmu obrysy a parametr `freetype-opacity` nastavuje průhlednost v rozmezí 0–255. Velikost písma lze nastavit dvěma způsoby: buď relativně, kdy se velikost nastaví v závislosti na velikosti videa (hodnota 12 zpříkladu znamená „velké“ písmo), nebo absolutně parametrem `freetype-fontsize`, kde hodnota udává výšku písma v obrazových bodech. Pro podrobnosti viz `vlc -p freetype --advanced`.

Obrazové filtry

Pro dodatečné úpravy obrazu i zvuku lze při přehrávání aplikovat filtry. Asi nikoho nepřekvapí úprava jasu a kontrastu obrazu či odstranění prokládání – některé vychytávky však jen tak jinde nenajdete. Filtry ve VLC vykonávají samostatné moduly, proto kromě nastavování parametrů těchto filtrů je také zapotřebí přehrávači sdělit, které moduly chceme používat. Slouží k tomu parametry `vout-filter` pro obrazové a `audio-filter` pro zvukové filtry. Pro úpravu například kontrastu tedy:

```
vlc video.mov --vout-filter adjust --contrast 1.2
```

Při použití více modulů od sebe jejich jména oddělíme dvojtečkou. A které ty moduly že to jsou?

adjust

Úprava obrazu – jas, kontrast, zabarvení, sytost a index gamma:

```
vlc video.avi --vout-filter adjust --contrast 1.2 --brightness 0.8 \\
--hue 67 --saturation 2.4 --gamma 1.33
```

Pro rozsahy hodnot viz `vlc -p adjust`.

deinterlace

Odstranění prokládání řádků, nepostradatelné zejména u televizního vysílání:

```
vlc dvb: --vout-filter deinterlace --deinterlace-mode mean
```

Možné režimy odstranění prokládání jsou `blend`, `bob`, `discard`, `linear`, `mean` a `x`.

invert

Inverze barev, tj. výsledkem je negativ.

transform

Už se vám někdy podařilo natočit video orientované na výšku, tj. otočené o devadesát stupňů? Tento filtr umožní se na něj podívat, aniž by bylo zapotřebí otáčet monitor:

```
vlc video.avi--vout-filter transform --transform-type 90
```

Kromě zadání úhlu – `90`, `180`, `270` – lze obraz otočit též podle horizontální a vertikální osy, hodnoty zadáváme jako `hflip` a `vflip`.

distort

Filtr pro deformaci obrazu nabízí pár možností, které určitě stojí za vyzkoušení. Obraz se může vlnit, odrážet ve vlnící se vodní hladině či simulovat komiksovou tvorbu:

```
vlc video.avi --vout-filter distort --distort-mode gradient
```

Hodnoty pro deformační režim jsou `wave`, `ripple`, `gradient`, `edge`, `hough` a `psychedelic`, pro první čtyři viz obrázky.

Dlužno podotknout, že filtry z tohoto modulu jsou poměrně náročné na výkon počítače.

crop

Ořezání obrazu; hodnota parametru `crop-geometry` se udává jako šířka × výška výsledného obrazu plus odskok od levého okraje plus odskok od horního okraje:

```
vlc video.mpg --vout-filter crop --crop-geometry 320x200+40+50
```

Alternativně lze ořez nastavit parametrem `autocrop`, který si vynutí automatické rozpoznání a oříznutí černých okrajů.

clone

Filtr pro klonování obrazu – zdrojové video si najednou můžeme pustit ve více oknech zároveň, přičemž každé z nich může využívat jiný výstupní modul:

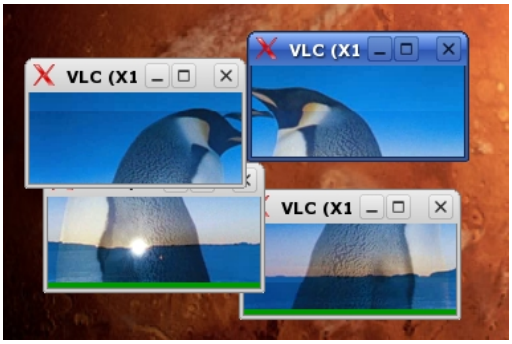
```
vlc video.mov --vout-filter clone --clone-vout-list aa,glx --clone-count 2
```

Parametr `clone-count` udává počet oken a `clone-vout-list` seznam výstupních modulů. Uvedeném příkladě se video bude přehrávat ve dvou oknech, jednou za použití *OpenGL* a podruhé jako *Ascii Art*.

wall

Video je pro změnu rozřezáno do více oken – napadá někoho praktické využití?

```
vlc trailer.mov --vout-filter wall --wall-rows 2 --wall-cols 2
```



Parametry udávají, na kolik *jakoby* řádků a sloupců se má video rozdělit.

equalizer

Příklad zvukového filtru – klasický ekvalizér se sadou přednastavených prostředí k výběru:

```
vlc hudba.mp3 --audio-filter equalizer --equalizer-preset headphones
```

Mezi možnými režimy nechybí `pop`, `rock`, `classical` či `techno`, kompletní seznam se vypíše na povel `vlc -p equalizer`.

Samostatnou kapitolu tvoří filtry, které umožňují do streamovaného video přidávat vlastní texty a logo – a základům síťového vysílání se bude věnovat již příští díl seriálu.

⋆⋆⋆

Jazyky a překladače – 2 (regulární výrazy a konečné automaty)

Michal Vyskočil

Jak jsme si ukázali v minulém díle, regulární jazyk je věta generovaná gramatikou typu 3 podle Chomského klasifikace gramatik – gramatikou regulární. V dnešním díle si poodhalíme tajemství těchto mocných nástrojů. Rozdílem oproti minulému dílu bude méně teorie a více praxe.

Webová verze článku obsahuje JavaScript. Pozn. ed.

Co je to regulární výraz

Na regulární výrazy existuje několik pohledů. Pro ty, kteří je používat neumí, se jedná o magickou sekvenci znaků, které se nikdy nesnaží porozumět. Pro ty, kteří je používají, se jedná o mocný nástroj pro práci s textem, který lze použít při hledání, pro úpravy, nebo pro ověření vstupního řetězce. No a pro ty, kteří něco vědí o teoretické informatice, se jedná o větu jazyka typu 3. Navíc ta poslední skupina lidí tuší něco o konečném automatu a jeho vztahu k regulárním jazykům, výrazům.

Pochopitelně se nám tu vynořuje otázka. Pokud jsou regulární výrazy výsledkem gramatiky typu 3, to znamená nejméně mocné formy, proč se jimi vůbec zabývat? Proč nenasadit rovnou prostředky klasického programovacího jazyka, který je nepochybně mocnější a zvládne více věcí? Důvodů je několik. Předně síla jazyků typu 3 na většinu úloh stačí (a navíc současné výrazy u moderních nástrojů už patří do vyšších typů). Druhým důvodem je stručnost a (ne)přehlednost zápisu, oproti programové implementaci (jak se přesvědčíte dále). Kniha Art Of Unix Programming [1] se v osmé kapitole [2] zabývá minijazyky pro řešení specializovaných úloh, mezi něž se řadí i regulární výrazy. Obvykle je jednodušší napsat regulární výraz, než vytvořit jeho programovou implementaci.

Implementace regulárních výrazů

Snad každého někdy napadlo – *Jak to ten* `grep` *(*`sed`*,* `perl`*, ...) vlastně dělá? Jaká je posloupnost kroků od zadaného výrazu až po kód, který jej provádí?* Vezměme jednoduchý výraz `x*y+`, který značí vezmi libovolný počet znaků x, následovaný alespoň jedním znakem y. Člověk, který o regulárních výrazech neslyšel, by pravděpodobně začal daný problém řešit asi takto:

```
function hledej1(inp)
{
byloX = false;      // promenne udavajici
byloY = false;      // stav resene ulohy
zac = 0;
kon = 0;
for (i = 0; i != inp.length; i++) {
ch = inp.charAt(i);
if (!byloX && !byloY && ch != "x" && ch != "y") { continue; }
if (!byloX && !byloY && ch == "x") { byloX = true; zac = i; }
if (!byloX && !byloY && ch == "y") { byloY = true; zac = i; }
if ( byloX && !byloY && ch == "y") { byloY = true; }
if ( byloX &&  byloY && ch != "y") { break;}
kon++;
}
msg1 = "hledej1(\""+inp+"\") --> x*y+ ";
msg3 = "\n";
if (!byloY) { msg2 = "nenalezen"; }
```

```
    else        { msg2 = "nalezen (" + zac + ", " + kon + ")";}
    return msg1+msg2+msg3;
    }
```

Dlouho jsem přemýšlel, jaký jazyk zvolit pro příklady – C, C++, Javu, Python? Nakonec jsem se rozhodl pro Javascript a to z toho důvodu, že všechny příklady jsou jednoduché a navíc mohou být spuštěny přímo v prohlížeči. Bohužel mi dalo trochu práce ladění, protože se zdá, že některé konstrukce v jistých prohlížečích nepracují, jako třeba `foreach` (Opera, MSIE) a indexování řetězců (MSIE). Nicméně skripty jsou úspěšně otestovány v prohlížečích Mozilla Firefox 1.5, Internet Explorer 6.0 a Konqueror 3.5 a Opera 9.0. Výsledek (dynamicky generovaný Javascriptem) máte zde:

```
hledej1("x") --> x*y+ nenalezen
hledej1("y") --> x*y+ nalezen (0, 1)
hledej1("xy") --> x*y+ nalezen (0, 2)
hledej1("xxyy") --> x*y+ nalezen (0, 4)
hledej1("axyb") --> x*y+ nalezen (1, 2)
hledej1("ayxb") --> x*y+ nalezen (1, 3)
```

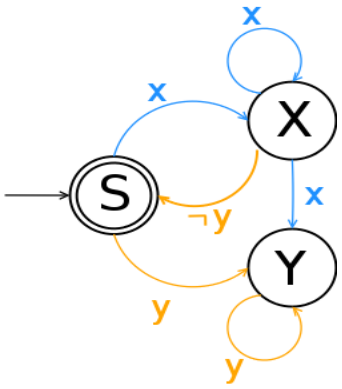
Zde je interaktivní prvek. Pozn. ed.

Je nutné poznamenat, že tento způsob implementace je velice (velice jemně řečeno) nevhodný. Kód je zmatený, složitý, špatně se chápe a tím poskytuje prostor pro chyby (ostatně moje implementace pracuje špatně). Navíc, byť i jednoduchá změna, znamená, že je nutné celý program změnit.

Konečný automat

Tím, co kód znepřehledňuje nejvíc, jsou stavové proměnné. Respektive jejich složité vztahy, které musíme v programu uhlídat pomocí složitých podmínek. Pokud se nám v kódu (v lepším případě v návrhu) podobné věci množí, bývá konečný automat nejvhodnějším řešením. Formálně je (nedeterministický, bude vysvětleno dále) konečný automat pětice $M = (Q, \Sigma, \delta, q_0 \in Q, F \subset Q)$

- Q je konečná neprázdná množina stavů
- Σ je konečná neprázdná vstupní abeceda
- δ je přechodová funkce (množina pravidel, které popisují přechody mezi stavy ve tvaru $\delta : \ Q \times \Sigma \rightarrow 2^Q$)
- $q_0 \in Q$ je počáteční stav
- $Fu\ Q$ je množina koncových stavů



Nejdůležitějším pojmem u konečného automatu je *stav*. Koneckonců se někdy nazývají stavovými automaty (Finite State Automata v angličtině). Automat lze graficky znázornit:

Jak vidíte, automat se sestává se 3 stavů S, X a Y, přičemž S je startovacím stavem a Y koncovým. Ne náhodou je tento automat ekvivalentní regulárnímu výrazy `x*y+`. Ve startovacím stavu S automat čeká na některý ze znaků x, nebo y. V případě příchodu znaku x se přepne do stavu X a ten neopustí, dokud nenačte znak y. Potom se přepne do stavu koncového stavu Y. Znak ¬y značí cokoliv mimo y, takže se ze stavu X můžeme dostat do počátku.

Praktická implementace by mohla vypadat takto (přidal jsem 4. stav F, abych nemusel používat `goto`, nebo cyklus `while`):

```
function hledej2(inp)
{
states  = {
```

```
        S : 0,
        X : 1,
        Y : 2,
        F : 3
    }
    st = states.S;
    zac = kon = 0;

    for (i = 0; i != inp.length; i++) {
        ch = inp.charAt(i);
        switch (st) {   case states.S:
            if (ch == "x") { st  = states.X; zac = i;}
            if (ch == "y") { st  = states.Y; zac = i;}
            break;
        case states.X:
            if (ch == "y")      { st  = states.Y; }
            else if (ch != "x") { st  = states.S; }
            break;
        case states.Y:
            if (ch != "y")      { kon = i - 1; st = states.F;}
            break;
        default:
            break;
        };
        kon++;
        if (st == states.F) { st = states.Y; break;}
    }

    msg1 = "hledej2(\""+inp+"\") --> x*y+ ";
    msg3 = "\n";
    if (st != states.Y)     { msg2 = "nenalezen";}
    else                    { msg2 = "nalezen (" + zac + ", " + kon + ")";}
    return msg1+msg2+msg3;
}
```

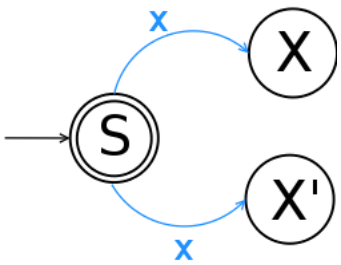
```
    hledej1("x") --> x*y+ nenalezen
    hledej1("y") --> x*y+ nalezen (0, 1)
    hledej1("xy") --> x*y+ nalezen (0, 2)
    hledej1("xxyy") --> x*y+ nalezen (0, 4)
    hledej1("axyb") --> x*y+ nalezen (1, 2)
    hledej1("ayxb") --> x*y+ nalezen (1, 3)
```

Stav	S	X	Y
x		X	X
y		Y	Y
...			

Tato, byť na počet řádků delší, ale jednoznačně přehlednější implementace konečného automatu, je přesně to, co mnoho programátorů s úspěchem používá. Je nutné říct, že takto *natvrdo* naprogramovaný stroj trpí špatnou rozšiřitelností, ale místo konstrukce `switch case` lze z výhodou použít hashovací pole. Ovšem i toto řešení trpí malou obecností. Pokud chceme změnit regulární výraz, nezbude nám, než kód přeprogramovat (anebo použít dynamické jazyky a hashovací pole místo konstrukce `case`). Ovšem je tu ještě jedna možnost – jiný pohled na konečné automaty a to tabulka přechodů. Prakticky stejně jsou implementovány všechny pomocné knihovny pro konečné automaty.

O algoritmech převodu

Ještě než budeme pokračovat, zbývá nám vysvětlit jeden pojem. Deterministický (DKA), versus nedeterministický automat (NKA). U DKA je přechod jednoznačně dán příští stav aktuálním stavem a podmínkou. NKA může mít na jednu podmínku následujících stavů více. Oba automaty jsou ekvivalentní, ale nedeterministické jsou rychlejší, protože nemusí prohledávat všechny možnosti a vyberou si vždy správnou větev. Nevýhodou je, že naše počítače, jak je známe dnes, jsou deterministické, což znamená, že nedeterministické stroje se stejně musejí převést na deterministické. Fragment NKA je zobrazen na následujícím obrázku.



Mluvím o tom z toho důvodu, že běžně používaný algoritmus převádí regulární výrazy právě na NKA. Pokud se narazí na větev, která vede na více možností, program začne prozkoumávat jednotlivé cílové stavy (backtracking). Klasický postup skončí v případě nálezu prvního vyhovujícího výrazu. Posixová varianta v případě použité konstrukce „nebo“ najde všechny vyhovující a vrátí nejdelší řetězec, ale tuto variantu běžné nástroje neimplementují, protože je nejpomalejší. Druhou možností je paralelní provádění všech cest, díky čemuž stačí jeden průchod a není třeba se vracet. Nevýhodou je skutečnost, že si nemůžete zapamatovat podvýrazy.

Závěr

Toto byl pouze lehký úvod do problematiky konečných automatů. Zcela jsem vynechal problematiku stavových akcí, pojmy jako Mealy a Moorův automat. Nemluvili jsme o greedy regulárních výrazech a podobně. Navíc jsem tvorbu stroje pro provádění regulárních výrazů lehce odbyl. Jednak dnešní nástroje obecně regulární výrazy podporují. A v případě, že ne, je nejlepším způsobem jednoduše použít knihovnu PCRE [3] (Perl Compatible Regular Expressions), která je syntakticky i sémanticky kompatibilní s regulárními výrazy Perlu. Což je defakto norma, protože skutečná norma POSIX se používá méně a její regulární výrazy nejsou tolik silné. Tuto knihovnu používají open source projekty Exim (pro nějž byla původně napsána), Apache, PHP, Postfix, nebo KDE.

Pokud chcete automaty *vidět* na vlastní oči, podívejte se na Animace Teoretické informatiky [4], jedná se o flashové animace konečných (DKA i NKA) automatů, převodů NKA na DKA a mnoho dalších zajímavých věcí. V dalším díle se zvolna přesuneme k problematice překladačů, na řadě je totiž syntaxe.

Odkazy

[1] http://www.faqs.org/docs/artu/
[2] http://www.faqs.org/docs/artu/minilanguageschapter.html
[3] http://www.pcre.org/
[4] http://www.cs.vsb.cz/jancar/TEORET-INF/ANIMACE/ti-animace.html

⋆⋆⋆

Jazyky a překladače – 3 (syntaxe)

Michal Vyskočil

Přesuneme se od regulárních výrazů trochu blíže k překladačům. Tématem dnešního dílu budou základy analýzy zdrojového kódu. Navíc přinese ještě méně teorie a více praxe než předchozí díl. Dnešním oblíbeným programovacím jazykem je C.

Princip překladače

Překladač je program, který (velice zjednodušeně řečeno) postupně prochází kód napsaný v nějakém programovacím jazyce a z něj generuje spustitelný soubor. Samotný překlad se sestává z několika fází:

Analýza zdrojového kódu

```
/* program v C */
int   n = 10;
char* s = "Hello, world!";
if (n == 10) {
    printf("%s\n", s);
}

" program ve Smalltalku "
n := 10.
s := 'Hello, world!'.
[n = 10] if:
    [ Transcript show: n, s; cr. ]

<!-- dokument v XML -->
<servlet>
    <servlet-name>webdav</servlet-name>
    <servlet-class>org.apache.catalina.servlets.WebdavServlet</servlet-class>
    <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
    </init-param>
</servlet>
```

Nahoře jsou ukázky zdrojového kódu napsaného ve dvou různých programovacích jazycích – C a Smalltalk (Squeak). A navíc příklad dokumentu ve formátu XML (protože syntaxe se netýká jen programovacích jazyků). Na první pohled je vidět jejich syntaktická odlišnost, přestože v konečném výsledku dělají to samé. Pokud vynecháme absenci typování ve Smalltalku, potom vidíme odlišné příkazy pro přiřazení (“`=`” vs. “`:=`”), jiné znaky pro ukončení příkazu (“`;`” vs. “`.`”), jiné komentáře a podobně. Navíc vidíme další rozdíl. Zatímco řetězec `if` je v C kódu zvýrazněn, ve Smalltalku ne. Není to překlep při psaní, ale zdůraznění skutečnosti, že `if` není ve Smalltalku *klíčové* slovo jazyka, ale normální metoda. Syntaxe tudíž určuje (mimo jiné) tvar základních konstrukcí jazyka a překladač (interpret, parser) je musí nějakým způsobem rozpoznat.

Opět se zaměříme na otázku, jak něco takového naprogramovat. Standardní knihovna jazyka C (ostatní jazyky mají ekvivalentní nástroje) poskytuje funkce pro práci se soubory, které jsou definovány v souboru stdio.h. Pomocí těch nejjednodušších můžeme na zdrojový kód nahlížet jako na jednorozměrné pole znaků – `'i','n','t',' ', ...`, případně jako na izolované řádky:

```
21:  int    n = 10;
22:  char*  s = "Hello world!";
```

Ovšem funkce jako `scanf()` už pracují na vyšší úrovni a umožňují zadat formátovací řetězec, podle něhož bude vstup přiřazen do proměnných. Potom můžeme napsat něco jako:

```
char type[10], name[10];
int value;

scanf("%s %s = %d", &type, &name, &value);
printf("type: %s, name: %s, value: %d\n", type, name, value);
return 0;
```

a přeložený program vrátí

```
$ echo "int n = 10;" | ./scanf
type: int, name: n, value: 10
```

Funkce `scanf` bohužel neumí dvě věci. Tou první je definovat vlastní parametry (například pro klíčová slova jazyka) a tou druhou je fakt, že jí není možné říct *čekej na libovolný ze zadaných vstupů*. To, co potřebujeme, je *něco*, co se dokáže na program podívat asi takto:

```
(INT int) (IDENTIFIER n) (=) (CONSTANT 10) (;)
```

Lexikální analýza a program (f)lex

To *něco* se odborně nazývá *lexikální analyzátor*. Ten, který dokáže zdrojový program rozdělit na jednotlivé *lexémy* (tokeny). Lexém je nejmenší logická jednotka v kódu programu, například identifikátor nebo operátory, případně ukončovací znak. Jsou definovány jako regulární výrazy, třeba identifikátor je v C definován takto – `[a-zA-Z_][a-zA-Z0-9_]*`. Což znamená, že prvním znakem může být libovolný znak latinské abecedy a podtržítko, následovaný libovolným počtem znaků latinské abecedy, arabských číslic a podtržítek. Lexikální analyzátor tedy načítá vstupní text a zkoumá, do které ze zadaných kategorií patří.

Přestože je možné napsat daný parser ručně (například jako velký konečný automat [1]), bývá jednodušší využít generátor analyzátorů. Program `lex` je dostupný na všech systémech, které splňují normu POSIX, a jeho úkolem je na základě symbolického popisu vygenerovat kód lexikálního analyzátoru. Ve většině linuxových distribucí ovšem naleznete progam `flex`, což je rozšířená verze `lex`u, kterou vydalo FSF. Specifikace skeneru se obecně sestává ze čtyř částí. V té první, ohraničené `%{ %}`, uvedeme deklarace jazyka C. Například makra, nebo hlavičkové soubory.

```
%{
#define UNDEFINED   0
#define HCONSTANT   1
#define OCONSTANT   2
#define DCONSTANT   3
%}
```

Ve druhé části si můžeme nadefinovat užitečné podvýrazy, které nám zpřehlední zápisy v další části.

```
D           [0-9]
O           [0-7]
H           [a-fA-F0-9]
IS          (u|U|l|L)*
```

Další část, ohraničená `%% %%`, slouží k definici lexémů. K nim může být přiřazen kód, který bude vykonán po jeho nalezení. Jak vidíte, lexémy jsou definovány pomocí regulárních výrazů a celý zápis trochu připomíná program `awk`. Zajímavostí je zápis `[ \t\n]+`, který nemá přiřazen žádnou akci. Je to z toho důvodu,

že bílé znaky (mezera, tabelátor, nový řádek) vynecháváme. Nicméně nástroj `make` nebo jazyk Python počítají i bílé znaky do syntaxe.

```
%%
0[xX]{H}+{IS}?   { return HCONSTANT; }
0{O}+{IS}?       { return OCONSTANT; }
{D}+{IS}?        { return DCONSTANT; }
[ \t\n]+
.                { return UNDEFINED; }
%%
```

Poslední, čtvrtá, část obsahuje již jenom výkonný kód v C, který reaguje na jednotlivé lexémy.

```
static char name[4][16] = {
    "nezname", "sestnactkove", "osmickove", "desitkove"
};

int main() {
    int token;
    while (token = yylex()) {
    printf("Nalezeno %s cislo\n", name[token]);
    }
}
```

K dispozici je zdrojový kód integer.l [2], i ve verzi se zvýrazněnou syntaxí integer.l.html [3]. Nebo si můžete stáhnout celý archív integer.tar.gz [4], který obsahuje i Makefile. Překlad se potom provede (pokud jste si nestáhli archív s Makefile) takto:

```
flex integer.l
gcc -o integer lex.yy.c -lfl
```

Aplikace potom načítá standardní vstup a analyzuje zadané lexémy:

```
$ ./integer
123
Nalezeno desitkove cislo
0123
Nalezeno osmickove cislo
0x115
Nalezeno sestnactkove cislo
0X1Ff
Nalezeno sestnactkove cislo
234U
Nalezeno desitkove cislo
12      0x2
        077
Nalezeno desitkove cislo
Nalezeno sestnactkove cislo
Nalezeno osmickove cislo
54,21
Nalezeno desitkove cislo
```

Po zadání znaku, který nezná `‘‘,’’`, se program ukončí, protože makro `UNDEFINED` se expanduje na 0, a tím dojde k ukončení cyklu `while`. Navíc také úspěšně ignoruje prázdné řádky. Vygenerovaný program obsahuje několik důležitých funkcí a proměnných:

`yylex()`

Funkce, která v cyklu spouští akce na základě nalezené masky. Protože ji přerušujeme příkazem `return`, můžeme ji použít v našem cyklu.

`yytext`

Pole znaků, které obsahuje nalezený lexém.

`yyleng`

Délka řetězce v `yytext`.

`yymore()`

Načte další lexém do proměnné `yytext`.

`ECHO`

Zkopíruje nalezený lexém na standardní výstup.

Zajímavosti

Manuálů k program flex naleznete na internetu mnoho, některé z nich jsou i dole v odkazech, proto nebudu vysvětlovat formáty masek nebo funkce přepínačů, ale přeskočím na zajímavější funkce.

Zkratky programu flex

V předchozím případě bylo nutné definovat podvýraz pro čísla jako `D [0-9]`. Autoři programu `flex` do něj zařadili zkratky pro často používané skupiny. Ty se zapisují ve formátu `[:zkratka:]`, což je stejná syntaxe, jako například v programu `grep`. Podporované zkratky jsou:

```
alnum, alpha, blank, cntrl, digit, graph, lower, print, punct, space, upper, xdigit
```

Programátorům v C budou jistě velice povědomé, protože názvy i funkčnost odpovídá fcím `is<zkratka>()` ze souboru ctype.h. Uvedený příklad potom můžeme zapsat jako `D [[:digit:]]` (a přiznám se, že teď dostalo slovo zkratka zcela nový význam).

Vstup a výstup

Někdy nám nemusí být po chuti, že analyzátor pracuje jen se standardním vstupem a výstupem. Jsou definovány dvě proměnné:

```
FILE *yyin  = stdin;
FILE *yyout = stdout;
```

Pokud v naší funkci `main()` napíšeme

```
yyin  = fopen("input", 'r');
yylex();
```

Při tom je důležité vědět, že existuje funkce `yywrap()` z knihovny `libfl`, která se používá k rozhodování, zda je analyzátor na konci. Můžete si ovšem napsat vlastní verzi této funkce, která bude nastavovat proměnnou `yyin` na základě nějakého seznamu, a tím zajistíte zpracování více než jednoho souboru (což je výchozí chování).

Generování analyzátorů v C++

Časy se mění a tak i do tradičních vod unixových nástrojů pronikl jazyk C++. Zároveň s tím se objevil požadavek na generátor analyzátorů pro tento jazyk. Dalo by se říct, že flex podporuje dva způsoby, jak toho dosáhnout.

Prvním a ne zcela čistým způsobem je napsat kód v C++, soubor vygenerovat a nakonec doufat, že půjde přeložit překladačem C++. Přitom je nutné mít na paměti, že řetězce jsou pouze pole znaků, nebo

že `yyin` není proud z C++, ale klasický soubor z C. Nicméně v dobách, kdy překladače (a standardní knihovna C++) nestály za nic, toto bylo asi rozumné řešení.

Dnes je lepší použít přepínač `-+` (případně wrapper `flex++`), který vygeneruje *čistokrevný* C++ kód. Důležité je potom rozhraní dvou tříd definované v souboru FlexLexer.h – FlexLexer a jeho potomek yyFlexLexer. Funkce a globální proměnné, které jsme používali v C verzi, se potom změní na metody jedné z těchto dvou tříd. Bohužel řetězce jsou pořád definovány jako pole znaků a ne jako `std::string`. Funkce `main()` potom může vypadat přibližně takto:

```
int main()
{
    FlexLexer* lexer = new yyFlexLexer;
    while(lexer->yylex() != 0);
    return 0;
}
```

Nevýhody

Nevýhoda je jedna: vygenerovaný analyzátor je pomalejší a náročnější na strojové prostředky než ručně napsaný protějšek. Navíc s pomocí konečného automatu není napsání analyzátoru nic těžkého. Na druhou stranu je definice analyzátoru kompaktnější a snadněji udržovatelná. Velice pěkně to ilustruje příklad na wikipedii (Lexical analysis [5]), kde je ručně napsaný analyzátor jazyka PL/0 (který stvořil Niclaus Wirth, takže odpůrci Pascalu byli varováni) a jeho protějšek v lexu. Asi 100 řádků ručně napsaného kódu oproti 50 řádkům v `lex`u, které se ovšem rozgenerují na asi 1700 řádků v C (nebo 1500 v C++).

Závěr

Dnes jsme se stručně seznámili s klasickým unixovým nástrojem `(f)lex`, který slouží ke generování lexikálních analyzátorů. Programů, které rozpoznávají *kousky* (lexémy, tokeny) syntaxe programovacího jazyka. Poznali jsme jeho výhody (rychlost tvorby, snadná udržovatelnost), ale i nevýhody (malý výkon, větší spotřeba systémových prostředků). Dnešní díl byl zaměřen na programátory v C (a trochu v C++) a aby to ostatním nebylo líto, přidám odkazy na obdobné nástroje pro ostatní jazyky.

JLex [6] – generátor analyzátorů pro Javu; JFlex [7] – rychlý generátor analyzátorů pro Javu; shlex [8] – modul pro psaní jednoduchých analyzátorů v Pythonu; Plex [9] – generátor analyzátorů pro Python; Cslex [10] – generátor pro C#; Parse::Lex [11] – modul pro jazyk Perl; lexer [12] – balíček lexer pro Common Lisp.

Odkazy

[1] www.abclinuxu.cz/clanky/programovani/jazyky-a-prekladace-2-regularni-vyrazy-a-konecne-automaty
[2] http://www.abclinuxu.cz/data/vyskocil/jazyky-a-prekladace-3-integer.l
[3] http://www.abclinuxu.cz/data/vyskocil/jazyky-a-prekladace-3-integer.l.html
[4] http://www.abclinuxu.cz/data/vyskocil/jazyky-a-prekladace-3-integer.tar.gz
[5] http://en.wikipedia.org/wiki/Lexical_analysis
[6] http://www.cs.princeton.edu/~appel/modern/java/JLex/
[7] http://www.jflex.de/
[8] http://docs.python.org/lib/module-shlex.html
[9] http://www.cosc.canterbury.ac.nz/~greg/python/Plex/
[10] http://www.zbrad.net/DotNet/Lex/Lex.htm
[11] http://search.cpan.org/~pverd/ParseLex-2.15/lib/Parse/Lex.pm
[12] http://www.geocities.com/mparker762/clawk.html#lexer

⋆⋆⋆

Jazyky a překladače – 4 (syntaxe 2)

Michal Vyskočil

V předcházejícím díle jsme se seznámili s lexikální analýzou. Ovšem lexikální analyzátory pracují na úrovni jednotlivých lexémů v jazyce. V dnešním díle postoupíme o krok dál a budeme se zabývat gramatikami a parsery.

Princip překladače

Překladač je program, který (velice zjednodušeně řečeno) postupně prochází kód napsaný v nějakém programovacím jazyce a z něj generuje spustitelný soubor. Samotný překlad se sestává z několika fází, z nichž druhá se nazývá:

Syntaktická analýza (parsing) zdrojového kódu

```
/* program v C */
int   n = 10;
char* s = "Hello, world!";
if (n == 10) {
    printf("%s\n", s);
}

" program ve Smalltalku "
n := 10.
s := 'Hello, world!'.
[n = 10] if:
    [ Transcript show: n, s; cr. ]
```

Nahoře jsou ukázky zdrojového kódu napsaného ve dvou různých programovacích jazycích – C a Smalltalk (Squeak). Jsou to přesně ty samé, které byly uvedeny v minulém díle [1]. Pokud máte zkušenosti s programováním, tak víte, že jazyky očekávají lexémy na přesně definovaných místech (snad s výjimkou jazyka Perl, který má nejvolnější pravidla, co znám :-). Což je úkol, na který už lexikální analyzátory nestačí, protože ty na zápise

```
printf("%s\n", s) if (n == 10);
```

neuvidí nic zvláštního. Přesto, jak už určitě tušíte, překladač jazyka C s tím souhlasit nebude a bude si stěžovat:

```
perl.c:7: error: syntax error before ?}? token
```

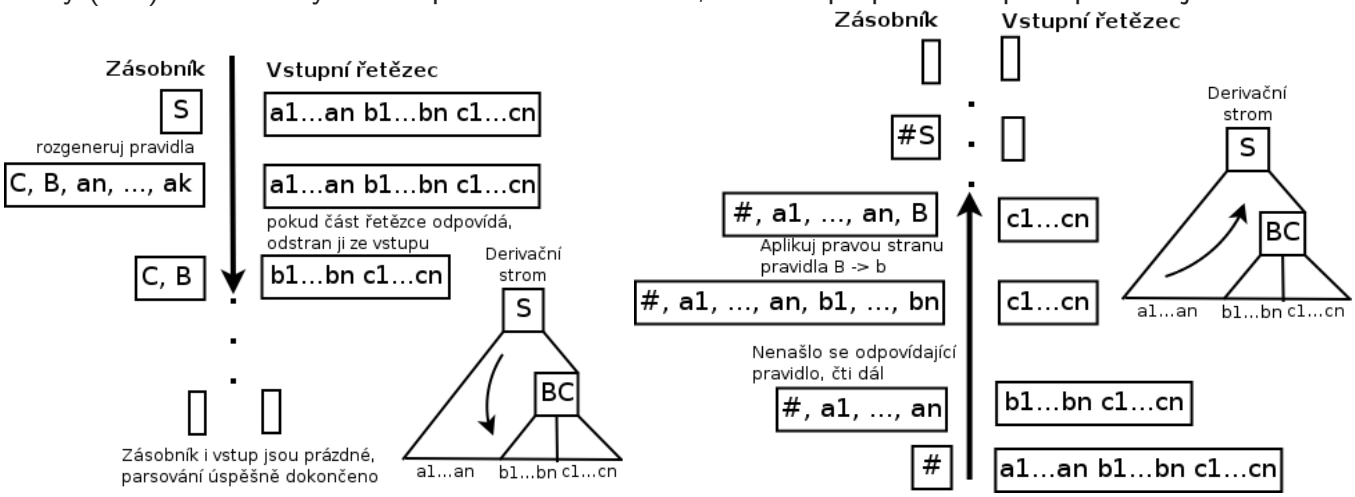
To znamená, že syntaxe je nejen množina lexémů jazyka, jak víme z minulého dílu, ale něco víc.

Gramatiky

Než se zaměříme na naši obvyklou implementační otázku, bude nutné opět udělat menší výlet do oblasti teorie. Pokud si pamatujete z prvního dílu [2], tak formální jazyky bývají generovány gramatikami. Programovací jazyky jsou zcela jistě jazyky formálními, to znamená, že mají svoje gramatiky. A právě ony zabezpečují část toho, co nazýváme slovem syntaxe. Například gramatika jazyka C přikazuje, že po klíčovém slovu `if` následuje výraz v závorkách následovaný buďto příkazem anebo skupinou příkazů (ohraničenou znaky `{}`). Zatímco gramatika jazyka Perl dovoluje, aby byla podmínka i za výrazem, gramatika jazyka C nikoliv. A protože se jí řídí jak programátor (většinou), tak i překladač (vždy, pokud nemá chybu), je možné, aby překladač dokázal zpracovat kód napsaný člověkem.

Gramatiky programovacích jazyků patří do třídy bezkontextových jazyků podle Chomského hierarchie. V této oblasti počítačoví vědci definovali několik podtříd, ke kterým patří jisté parsovací algoritmy. Pohled do příslušné kategorie na wikipedii (Category:Parsing algorithms [3]) nám dává představu o jejich počtu. Samotný proces, který nazýváme parsování (nebo syntaktická analýza), jsou postupné derivace řídící gramatiky (respektive tvorba derivačního stromu – parsing tree) a ověřování, zda jí lexémy odpovídají. Většina parserů potom používá jednu ze dvou strategií.

První z nich je *top-down* parsing (shora dolů). Tento parser pracuje tak, že vychází ze startovacího symbolu gramatiky a postupně hledá derivace. Ten se také často implementuje rekurzivním sestupem, kdy se pro každý (non)terminální symbol napíše obslužná funkce, které se při parsování postupně volají.



Druhá je *bottom-up* parsing (zdola nahoru). Ten naopak dělá postupně operace shift-reduce a pracuje od nejnižších symbolů gramatiky nahoru ke startovacímu.

Typy parserů

Obecně můžeme říct, že je gramatika určitého typu, pokud k ní můžeme sestrojit parser stejného typu.

LL parsery

LL parsery používají parsing shora dolů, zpracovávají vstup zleva doprava a konstruují nejlevější derivaci. Proto se také nazývá L (left-to-right) L(leftmost derivation). Občas se setkáváme s označením LL(k), kde *k* značí počet tokenů, které potřebujeme znát při rozhodování o průběhu další analýzy bez toho, aby bylo třeba používat backtracking. Také se v této souvislosti používá pojem *look-ahead*. Těmto parserům se říká evropské, mimo jiné i proto, že se jim věnoval Niclaus Wirth, který publikoval množství prací o optimalizaci LL(1) parserů. Wirthovo zaměření na rychlost překladu je jeden z důvodů, proč je překlad Pascalu obecně rychlejší, než překlad podobných jazyků. Rozhodně to platilo na 486 a překladačích společnosti Borland. Pascal se překládal bleskově, C už pomaleji (o C++ ani nemluvě). Prakticky do nedávné doby se tyto gramatiky příliš nepoužívaly, ovšem na počátku 90. let minulého století došlo ke změně přístupu. Vezměme jednoduchoučkou gramatiku s pravidly (pravidlo 3 není formálně správně zapsané, ale doufám, že mi bude odpuštěno, $\backslash d$ značí libovolné číslo).

1. S → N
2. S → (S + N)
3. N → $\backslash d$

Vidíme, že gramatika obsahuje terminální symboly `(`, `)`, $\backslash d$, `+` a navíc si doplníme symbol pro konec vstupu `$`. Potom bude parsovací tabulka (pravidla pro konstrukci přeskočím) vypadat

```
    (   )   \d  +   $
S   2   --  1   --  --
```

```
 N   --  --  3   --  --
```

Top-down parsing začíná ze startovacího symbolu, takže na počátku je v zásobníku

```
[ S, $ ]
```

Projdeme si výraz `(1 + 2)$`. Když parser požádá lexikální analyzátor o další token, získá znak `(`. V parsovací tabulce je pro symbol na vrcholu `S` a vstup `(` příkaz, použij pravidlo #2 (přičemž se v dalším kroku znak `(` ze zásobníku zahodí). Tím dostaneme

```
[ S, +, N, ), $ ]
```

V dalším kroku lexikální analyzátor vrátí 1, to znamená, že pro dvojici `[S, 1]` použijeme pravidlo #1 (nahrazení znaku S za N). Abychom si to zkrátili, rovnou i pravidlo #3 (nahrazení N za $\backslash d$).

```
[ \d, +, N, ), $ ]
```

Načtený symbol `1` odpovídá $\backslash d$, takže jej můžeme ze zásobníku odstranit. Další kroky jsou již analogické:

```
[ +, N, ), $ ]  // nacteno +, muzeme odstranit
[ N, ), $ ]     // nacteno 1, aplikujeme #3
[ \d, ), $ ]    // nacteno 1, muzeme odstranit
[ ), $ ]        // nacteno ), muzeme odstranit
[ $ ]           // nacteno $, muzeme odstranit
[ ]             // parsovani uspesne dokonceno
```

Tímto postupem jsme získali *nejlevější* derivaci – $S \rightarrow (S + N) \rightarrow (N + N) \rightarrow (\backslash d + N) \rightarrow (\backslash d + \backslash d)$

LR parsery

LR parsery potom logicky konstruují nejpravější derivaci, ale vstup zpracovávají zleva doprava. Používají parsing zespoda nahoru. Jejich výhodou je fakt, že je možné jejich parsery implementovat velice efektivně a mnoho používaných jazyků má LR gramatiku (jedna z výjimek je třeba Perl). Méně příjemné je to, že ruční konstrukce těchto parserů je obtížná a spousta lidí dává přednost generátorům analyzátorů. V závislosti na způsobu vytváření tabulky se tyto generované parsery dělí na SLR (*Simple LR*), LALR (*look-ahead LR*), Canonical LR a jiné.

SLR gramatiky jsou téměř identické s LR(0) gramatikami. Jedná se o nejjednodušší typ a pro mnoho jazyků je zcela nedostatečný. LALR parsery již dokáží zpracovat gramatiky LR(1) a velice dlouhou dobu byly oblíbené, protože dokáží vyjádřit více, než SLR gramatiky, přičemž je jejich parsovací tabulka rozumně velká. Algoritmus LR parseru (i jeho tabulka) je daleko složitější. Občas se jim říká *shift-reduce* a to podle operací posun a redukce, které obsahují. Jejich princip spočívá v tom, že postupně redukují vstupní řetězec *w* pomocí pravých stran derivačních pravidel tak, až dojdou ke startovacímu symbolu gramatiky. A dále je tu i GOTO tabulka, která zobrazuje přechody mezi jednotlivými stavy automatu. Zájemce odkáži na popis ve wikipedii (Architecture of LR parsers [4]).

Ostatní parsery

Earley parser (top-down) je pojmenován po svém tvůrci, kterým je Jay Earley. Tento parser dokáže zpracovat libovolnou bezkontextovou gramatiku. Jeho složitost je kvadratická až kubická, ovšem při použití tohoto způsobu parsování není třeba gramatiku nikterak *omezovat*. Nejlepší výsledky dává s gramatikami, které jsou zleva rekurzivní. Tento algoritmus implementují některé nástroje. *CYK*, neboli Cocke-Younger-Kasami parser (bottom-up). Jeho algoritmus je velice jednoduchý, zápis v pseudojazyce zabírá 14 řádků a jeho implementace v C++ (včetně pomocných operátorů) přibližně 200 řádků. Nejhorší asymptotická složitost je kubická, což znamená, že je srovnatelný s Earleyho algoritmem. Ve své základní podobě vyžaduje, aby byla gramatika zapsána v Chomského normální formě, ale dá se rozšířit tak, aby používala libovolně zapsanou gramatiku.

Forma gramatiky

Nejen pro sportovce, ale i pro gramatiky je důležitá forma. Před chvílí zmíněná Chomského normální forma znamená jistý způsob zápisu derivačních pravidel gramatiky, které ovšem přinášejí zajímavé vlastnosti. CNF je gramatika s pravidly ve tvaru

- $A \rightarrow BC$, nebo (A, B, C jsou nonterminální symboly)
- $A \rightarrow \alpha$, nebo (α je terminální symbol)
- $S \rightarrow \epsilon$ (volitelné pravidlo, S je startovací symbol)

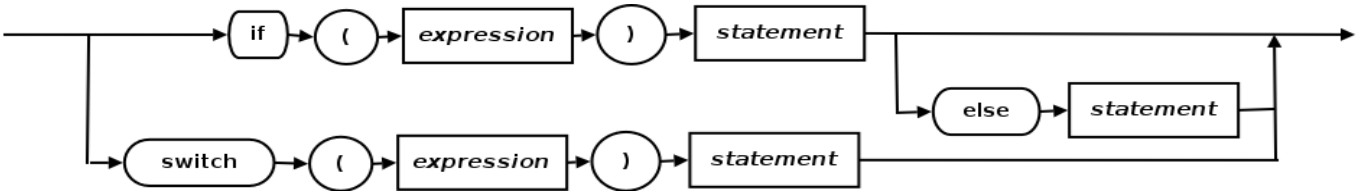
Dá se dokázat, že každá gramatika v Chomského normální formě je bezkontextová a navíc, že každá bezkontextová gramatika se dá převést na Chomského normální formu. Výhodou je, že při parsování produkuje binární strom, jehož výška je omezená velikostí největšího řetězce. Nebo fakt, že zpracování každého řetězce bude trvat 2n – 1 derivačních kroků, kde n je délka řetězce.

Ovšem daleko známější než Chomského forma, bude zcela určitě extended Backus Noam form – EBNF, která vznikla zjednodušením původní Backus-Noam formy (BNF). Například jazyk Algol-60 byl první jazykem, který byl popsán BNF. Tedy vznikla... vytvořil jí známý (a zde několikrát vzpomenutý) Niclaus Wirth. Postupem času se z ní stala norma ISO-14977. Skupina w3c používá upravenou verzi pro definici formátu XML a další varianta Augmented BNF se stala dopručením (RFC) číslo 4234 organizace IETF.

EBNF představuje často používaný způsob zápisu gramatik i proto, že jej mnoho lidí považuje za přehlednější než klasický zápis (nepotřebuje šipky a podobné speciální symboly). Navíc, pokud existuje k nějakému jazyku norma, bývá jeho syntaxe právě v této formě. Existuje jak grafická, tak i textová forma zápisu. Pro výše vzpomenutou podmínku `if` jazyka C existuje EBNF zápis

```
selection-statement:
    if, '(', expression, ')', statement, [ else, statement ] |
    switch, '(', expression, ')', statement
```

a jemu odpovídající grafická forma



Závěr

V tomto díle jsme si představili teorii, která se zabývá parsováním zdrojového kódu. To je také důvod, proč obyčejné regulární výrazy nemohou nikdy nahradit plnohodnotný parser. Regulární jazyk nikdy nemůže obsáhnout gramatiku napsanou v bezkontextovém jazyce. Je ovšem třeba poznamenat, že tohle se týká především jazyků algolského typu. Existují programovací jazyky, které se obejdou i bez složitě definované gramatiky se spoustou syntaxe a které můžeme obsáhnout regulárním jazykem. Ovšem ty bývají většinou programátorů považovány za *divné* a nepraktické. Příkladem může být Lisp, Fort anebo třeba Brainfuck (který je ovšem skutečně nepraktický). V příštím díle, který bude opět více praktický, si představíme generátory analyzátorů, protože je jejich ruční psaní nepochybně složitější než psaní lexikálního analyzátoru.

Odkazy

[1] http://www.abclinuxu.cz/clanky/programovani/jazyky-a-prekladace-3-syntaxe
[2] http://www.abclinuxu.cz/clanky/programovani/jazyky-a-prekladace-1-uvod
[3] http://en.wikipedia.org/wiki/Category:Parsing_algorithms
[4] http://en.wikipedia.org/wiki/LR_parser#Architecture_of_LR_Parsers

⋆⋆⋆

Beamer: LaTeX na prezentace

Petr Zelenka

„Potřeboval bych spáchat pěknou prezentaci, nevíš o nějakém programu?“ zeptal se Matěj. „No, o něčem bych věděl,“ odvětil na to Bohouš, „výsledek pak vypadá fakt dobře. Slyšel jsi už někdy něco o Beameru?“

LaTeX na prezentace? Proč ne!

Beamer [1] je – jednoduše řečeno – rozšiřující třída LaTeXu, kterou lze použít místo klasických tříd dokumentu typu `article` či `book`. Příprava prezentací je tak poněkud odlišná od „naklikám si někam, co potřebuji“. Způsob se totiž (překvapivě) shoduje s přípravou běžného dokumentu v LaTeXu.

To ale neznamená, že by práce byla zdlouhavější. Je prostě jiná a je třeba si na ni zvyknout. Věřím tomu, že vytvoření ekvivalentní prezentace třeba s OpenOffice.org Impress by mi zabralo daleko víc času. Hlavně proto, že s Impress neumím pracovat.

Beamer přenáší všechny výhody i nevýhody LaTeXu i do tvorby prezentací. Přidává možnosti vytvářet přechody mezi jednotlivými odrážkami, obsahuje řadu témat vhodných pro krátké i delší prezentace. Ale hlavně je distribuován s několika vzorovými soubory. Jejich editace pak umožní vytvořit pěknou prezentaci i těm, kteří s LaTeXem a/nebo Beamerem začínají.

Instalace

„LaTeX už jsem nainstalovaný měl, takže už zbýval jen ten Beamer.“ „A zvládl jsi to?“ byl Bohouš zvědavý. „Jasně, nejsem lama!“ potvrdil Matěj, „teď už chybí jen ta prezentace.“ Instalace by neměla představovat problém. Využijte buď možností vaší distribuce nebo stránek projektu, dokumentace je také výborná. V mém případě platila první varianta, Synaptic nezklamal a instalace proběhla bez zaváhání. Nutno podotknout, že LaTeXjsem již nainstalovaný měl.

Prezentace krok za krokem

„No tak jdeme na to. Jak dlouho budeš asi tak mluvit?“ „Mělo by to být tak na 20 minut, potom je tak 5 minut na otázky. Proč se ptáš?“ „Nejdřív musíme vybrat správnou šablonu,“ vysvětloval Bohouš, „Ne každá se hodí pro každou příležitost.“

Nejjednodušším způsobem, jak začít s prezentací, je vybrat již předpřipravený „polotovar“ a do něj doplnit náš text. Tyto „polotovary“ se nacházejí v adresáři s dokumentací, podadresáři solutions. V mém případě je to /usr/share/doc/latex-beamer/solutions. A protože začít se má od začátku, vytvoříme nejdřív úvodní snímek.

Úvodní snímek

„OK, takže téma je Technologie flash pamětí,“ pokračoval Bohouš, „To vysázíš příkazem `\title`*. Pro tebe to znamená zapsat to do složených závorek za tím. Můžeš přidat ještě kratší variantu pro případ, že by se nadpis někam nechtěl vejít, ale tenhle je krátký dost.“* Na úvodní snímek patří téma prezentace, jméno autora, datum a název školy či firmy, kterou přednášející reprezentuje (je-li nějaká). Mezi běžnými snímky prezentace má ten úvodní zvláštní postavení. Vytvoříme ho jediným příkazem – `titlepage` – uzavřeným do prostředí `frame`. Informace, které mají být zobrazeny, čerpá z hlavičky. Vše je nejlépe patrné z Matějovy prezentace, která zatím vypadá takto (matej01.tex [2], matej01.pdf [3]):

```
\documentclass{beamer}
```

```
\mode<presentation> {
  \usetheme{Warsaw}
  \setbeamercovered{transparent}
}

\usepackage{ucs}
\usepackage[utf8x]{inputenc}
\usepackage{czech}
\usepackage{palatino}
\usepackage{graphicx}

\title{Technologie flash pamětí}
\author{Matěj Novák}
\institute[UŠ JAK]{Ukázková škola Jana Ámose Komenského}
\date{23.~9.~2006}

\begin{document}

\begin{frame}
  \titlepage
\end{frame}

\end{document}
```

Pro překlad do `pdf` je možno použít buď `pdfcslatex`, nebo třeba trojici `cslatex`, `dvips` a `ps2pdf`. Zbývá ještě doplnit význam několika vybraných příkazů. Příkazem `\documentclass{beamer}` si řekneme o třídu dokumentu Beamer, která nám dá k dispozici příkazy jako například `\frame`. K dalším patří výběr tématu `\usetheme{Warsaw}` či příkaz `\usepackage`, kterým říkáme, jaké dodatečné balíky chceme použít (například `graphicx` dovolí vkládat obrázky příkazem `\includegraphics`).

Osnova

„Tak, titulní stránka by byla, teď už můžeme začít tvořit. Dobré je, když si nejdřív vytvoříš hrubou osnovu. To znamená na začátku nějaký rychlý úvod do tématu, případně motivaci. Pak hlavní část, kterou případně rozdělíš do smysluplných podkapitol. Na konci by nemělo chybět stručné shrnutí všeho, co jsi řekl.“

Osnova prezentace se obvykle umisťuje hned za titulní stránku a potom před každou nově začínající kapitolu v případě, že chceme, aby posluchač neztratil nit. Osnovu vysázíme tak, jak jsme zvyklí sázet v článku obsah. Snímek s osnovou tedy může vypadat například takto:

```
\begin{frame}
 \frametitle{Osnova}
 \tableofcontents[pausesections]
\end{frame}
```

Příkaz `\frametitle`, který jsme ještě neměli tu čest poznat, vysází (jak by se dalo čekat) nadpis snímku; `\tableofcontents[pausesections]` se pak postará o vysázení osnovy. Nepovinný parametr `pausesections` způsobí to, že se jednotlivé body osnovy (rozuměj teď hlavní body), budou odkrývat postupně, na klepnutí. Chceme-li zobrazit celou osnovu naráz, tento parametr smažeme. Protože je osnova generována opět jen jedním příkazem, zbývá dovysvětlit, kde bere text. Hlavní body osnovy tvoří parametr příkazů `\section` umístěných ve zdrojovém textu mezi jednotlivé snímky. Podbody jsou tvořeny obdobně, příkazem `\subsection`. Nejlépe je použití vidět na Matějově příkladu (matej02.tex [4]):

```
\begin{frame}
```

```
  \frametitle{Osnova}
  \tableofcontents
\end{frame}

\section{Co bylo cílem projektu}

\section{Technologie}

\subsection{NOR}

\subsection{NAND}

\subsection{Inovace}

\subsection{Kterou vybrat?}

\section{Náhled do budoucnosti}

\section*{Souhrn}
```

Jednotlivé snímky a co na ně

„Teď už můžeš do jednotlivých sekcí vkládat snímky a psát do nich text. Říká se, že k jednomu snímku se mluví v průměru tak minutu až dvě, ale je to dost individuální. To znamená, že tvoje prezentace by měla obsahovat zhruba 15 snímků.“

Text by měl být při prezentaci psán formou odrážek. Proto se při jejich tvorbě budeme často setkávat s výčtovými prostředími. Prvním z nich je `itemize`, které obsažené položky vysází s puntíkem (či jiným znakem) na začátku. Obdobou je v `html` tag `<ul>`.

```
\begin{itemize}
  \item položka seznamu
  \item další odrážka
\end{itemize}
```

Chceme-li na začátku místo nějaké značky čísla, použijeme prostředí `enumeration`, tzn. v předchozím případě jen nahradíme slůvko `itemize` slůvkem `enumeration`. To si lze představit jako `html` tag `<ol>`. Třetím prostředím do party je `description`, jehož pomocí lze vytvářet jakési slovníkové vysvětlivky.

```
\begin{description}
  \item[Strom] Graf, který neobsahuje kružnici.
\end{description}
```

Ke zvýraznění některého ze slov lze použít klasického postupu v LaTeXu, tedy uzavřít ho do `\emph`. Třída Beamer nám ovšem nabízí ještě jiný způsob, a to uzavření textu do `\alert`, při kterém bude tento text vysázen červeně. Pro zvýraznění celého bloku textu (například podmínek, za kterých něco platí), můžeme využít prostředí `block`, které svůj obsah vloží do hezkého rámečku. Takhle se rozhodl Matěj zdůraznit cíle svého projektu (matej03.tex [5], matej03.pdf [6]):

```
\section{Co bylo cílem projektu}

\begin{frame}
  \frametitle{Cíle projektu}
  \begin{block}{Cíle}
```

```
    \begin{itemize}
      \item Prozkoumat, jak pracuje paměť flash.
      \item Zdokumentovat současné technologie flash pamětí.
      \item Porovnat jednotlivé typy.
      \item Provést stručný rozbor, k~čemu se jednotlivé typy hodí.
    \end{itemize}
  \end{block}
\end{frame}
```

Takovýto blok má modré záhlaví s nápisem Cíle a namodralé tělo. Chceme-li být ještě důraznější, můžeme použít prostředí `alertblock`, které je laděno do červena. Obdobně pak fungují specializovaná prostředí pro definice (`definition`), teorémy (`theorem`), příklady (`example`) a jiné (`lemma`, `proof`, `corollary`).

„Hmm, tak to vypadá vážně dobře," odtušil spokojeně Matěj, „ale ještě třeba nevím, jak vložit obrázek nebo vytvořit tabulku." „Není to nijak složité," uklidnil ho Bohouš, „ale protože už budu musel běžet, necháme to na jindy."

Pokračování

Protože obrázek řekne často víc než tisíc slov, podíváme se příště na to, jak ho do naší prezentace dostat. Dále si ukážeme práci s tabulkou, vytváření snímků, na kterých se jednotlivé prvky postupně odkrývají, a třeba i něco navíc.

Odkazy

[1] http://latex-beamer.sourceforge.net
[2] http://www.abclinuxu.cz/data/zelenka-petr/beamer/matej01.tex
[3] http://www.abclinuxu.cz/data/zelenka-petr/beamer/matej01.pdf
[4] http://www.abclinuxu.cz//data/zelenka-petr/beamer/matej02.tex
[5] http://www.abclinuxu.cz/data/zelenka-petr/beamer/matej03.tex
[6] http://www.abclinuxu.cz/data/zelenka-petr/beamer/matej03.pdf

⋆ ⋆ ⋆

OSPF – dynamické routování

Dušan Hokův

Snad každý jel jednou autem a ví, že z místa A do místa B může jet jednou třeba po dálnici, jindy třeba přes místo C anebo objížďkou a musí jet po okreskách. Podobně se může chovat i paket při svém putování sítí.

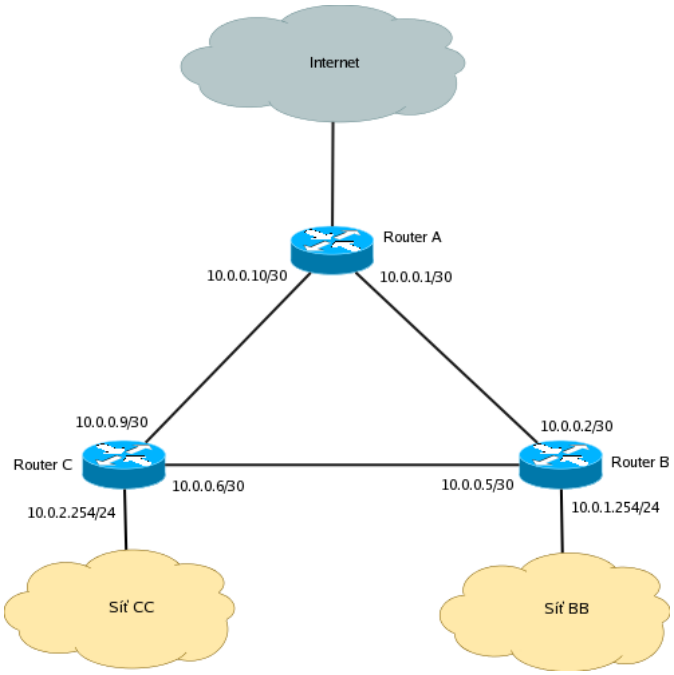
Cílem tohoto článku je vysvětlit, jak na to. V našem případě se jedná o kombinaci Linux, routovací démon quagga [1] a protokol OSPF. Není mou snahou dopodrobna ihned rozpitvat všechny pojmy, ale rád bych čtenářům vysvětlil, jak začít, a třeba postupně se i ke složitějším situacím a řešením dopracoval.

Protokol OSPF

Název znamená Open Shortest Path First a jedná se protokol používaný pro interní routování uvnitř autonomního systému (AS). Autonomním systémem může být např. infrastruktura ISP – linky po republice nebo v menším měřítku například wifi síť v nějaké lokalitě. OSPF a dynamické routování obecně se používá v případech, kdy se cesta ven (default route) dá uskutečnit více než jednou trasou. V praxi to znamená v podstatě ten příklad s jízdou autem.

OSPF je tzv. link-state protokol, pomocí kterého si sousední routery prostřednictvím hello paketů vyměňují informace o stavu linek, a každý z routerů v dané oblasti (area) zná celou topologii sítě. Výchozí konfigurace je *hello paket* jednou za 10 sekund, a pokud nepřijde žádná odezva do 40 sekund, vyprší *dead interval* a linka je prohlášena za mrtvou. Poté následuje přepočet a router hledá cestu jinudy, nezapomene však informovat všechny své sousedy (neighbor). Proces OSPF zjištěné informace předá démonu zebra, který již kernelu řekne novou default routu a routy přijaté od sousedů.

Příklad dynamického okruhu



Na obrázku je vidět situace, kdy router A je tzv. hraničním routerem v oblasti, routery B a C jsou routery v různých lokalitách této oblasti. Z obrázku rovněž vyplývají možné cesty ven. V tomto případě se tedy quagga postará o to, že router A bude propagovat ostatním routerům sebe jako default routu. Pomocí *cost* můžeme určit „cenu" linky a tím určit i preferovanou trasu, nebo můžeme traffic rovnoměrně rozprostřít (load balancing, equal cost multipath) a nechat jej téct více směry. Dokonce můžeme zařídit, že směr

ven půjde jinou cestou než směr dovnitř. V našem případě použijeme na všech spojnicích cost 100, tedy všechny tři linky jsou kvalitativně stejné. Neurčujeme žádnou preferovanou trasu. Platí princip, že čím menší cost, tím kvalitnější linka a má přednost. V případě poruchy linky např. mezi routerem A, B se tedy nastaví na routeru B default routa přes C s costem 201 a rovněž cost na router A bude 200. Na routeru A bude cost na router B 200 a trasa bude přes router C. Samozřejmě, že se vzájemně vypropagují i routy na sítě BB a CC. Pro všechny případy je nutné vypnout na routerech RP Filter, aby routerům nevadilo, že odpověď na paket odeslaný jedním rozhraním může přijít z jiného:

```
#!/bin/bash
for i in /proc/sys/net/ipv4/conf/*/rp_filter; do
echo "0" > $i; done
echo "1" > /proc/sys/net/ipv4/conf/lo/rp_filter
```

V tomto případě jsou mezi jednotlivými routery použity spojovací sítě /30 (4 adresy). Za routery B a C jsou v sítích BB a CC koncoví uživatelé, kteří mají default routu nastavenu na router B resp. C.

Konfigurace routerů

Na routerech B a C nenastavovat default routu. Ta bude nastavena právě díky OSPF podle výpočtu nejkratší trasy. O komunikaci s kernelem a mj. i nastavení default routy se přes rozhraní Netlink postará démon zebra. V tomto konfiguračním souboru je třeba nastavit název, heslo, heslo pro pokročilé funkce, vyjmenovat interface, případně vyjmenovat statické routy. Na hraničním routeru (v našem případě router A) by zde měla být i default routa směrem ven.

Příklad konfigurace zebra.conf

```
hostname router
password nejakeheslo
enable password nejakejineheslo
log file /dev/null
service advanced-vty
!debug zebra events
!debug zebra kernel
!
interface lo
!
interface eth0
!
interface eth1
!
interface eth2
!
```

Konfigurace OSPF démona je o něco složitější, ale nic záludného to není. Opět zde nastavíme hostname, obě hesla, poté se vyjmenují interface, na kterých se má OSPF používat, a nakonec globální nastavení včetně pokročilých nastavení.

Příklad konfigurace ospfd.conf

```
! Jmeno routeru
hostname router
! Heslo pro vzdaleny vty pristup (konzole)
password nejakeheslo
```

```
! Heslo pro prechod do rozsireneho modu v quagga shellu (konzoli)
enable password nejakejineheslo
! Kam logovat (doporucuji po odladeni /dev/null, nebot zebra a ospfd
! na disk chrli mega a mega logu)
log file /dev/null
! Povolit funkce rozsireneho virtual-terminalu (konzole)
service advanced-vty
!
! Co logovat
!debug ospf ism
!debug ospf zebra
!
! Loopback
interface lo
description system loopback
!
! Prvni sitovka zakomentovana, do internetu nebudeme vysilat OSPF
!interface eth0
!
! Druha sitovka
interface eth1
description smer router
A ip ospf cost 100
ip ospf dead-interval 40
ip ospf hello-interval 10
!
! Treti sitovka
interface eth2
description smer router C
ip ospf cost 100
ip ospf dead-interval 40
ip ospf hello-interval 10
!
!
! Konfigurace ospfd
router ospf
! ID-routeru (napr. IP adresa)
ospf router-id 10.10.10.1
redistribute connected metric-type 1
redistribute static metric-type 1
!
network 10.0.0.0/24 area 0
network 10.0.1.0/24 area 0
network 10.0.2.0/24 area 0
!
!tohle jen na hranicnim routeru (router A) -- propagovani default routy
default-information originate always metric-type 1
```

Díky těmto nastavením a OSPF se na router A vypropagují i statické routy na sítě BB a CC z routerů B a C a rovněž mezi B a C si vzájemně předají routy na BB a CC.

Důležité: Sousedi musí mít vzájemně shodné nastavení intervalů pro hello pakety a dead interval, jinak se spolu nedomluví. Rovněž router-id musí být v celé OSPF síti unikátní, jinak se ospf proces bude

chovat velice divně, pokud náhodou úplně nezkolabuje. Toto bývají jedny z nejčastějších chyb, proto si své konfigurační soubory důkladně přečtěte a možné překlepy opravte.

Routovací tabulka na routeru B

```
Adresát      Brána        Maska           Přízn Metrik Odkaz  Užt Rozhraní
10.0.0.0     0.0.0.0      255.255.255.252 U     0      0        0 eth0
10.0.0.4     0.0.0.0      255.255.255.252 U     0      0        0 eth1
10.0.2.0     10.0.0.6     255.255.255.0   UG    100    0        0 eth1
10.0.1.0     0.0.0.0      255.255.255.0   U     0      0        0 eth2
127.0.0.0    0.0.0.0      255.0.0.0       U     0      0        0 lo
0.0.0.0      10.0.0.1     0.0.0.0         UG    101    0        0 eth0
```

Pomocí metrik se vypočítává nejkratší cesta a z výpisu je poznat, že routy označené metrikou nastavila zebra. Ještě lépe je to vidět, použijete-li příkaz `ip route`, např. pro routu do sousední sítě:

```
10.0.2.0/24 via 10.0.0.6 dev eth1  proto zebra  metric 100
```

Quagga VTY shell

Démony zebra i ospfd lze během provozu ovládat přes konzoli (quagga shell, vty). Ve výše uvedeném nastavení budou poslouchat pouze na loopbacku. Přihlášení lze provést pomocí telnetu, přičemž zebra poslouchá na portu 2601 a ospfd na portu 2604. Podrobnostem se budu věnovat později.

Závěr

Kdo umí routovat staticky a chápe základní principy, tomu by nemělo dynamické routování pomocí OSPF činit vážnější potíže. To by bylo pro začátek vše, teď by bylo dobré trošku si zaexperimentovat, ať není jen teorie. To už je ale na každém z vás. Pokud bude zájem, rád bych se rozepsal podrobněji například o věcech jako je route-mapa, quagga vty shell, zabezpečení komunikace, propojení se snmp včetně posílání trapů a složitějších topologiích a situacích, včetně různých úskalí přechodu od statického routování k dynamickému.

Odkazy

[1] http://www.quagga.net

⋆ ⋆ ⋆

WYSIWYG editor online – 2 (formátovánie textu)

Matej 'Yin' Gagyi

V minulej časti sme si vytvorili jednoduchý textový editor priamo na stránke. Dnes ho rozšírime o funkcie formátovania textu a editor samotný zaradíme do formuláru na stránke. Tiež sa oboznámime s niekoľkými nedostatkami editora Midas, ktoré nie je jednoduché riešiť.

Formátovanie textu

Aktivovanie editora nad HTML dokumentom sprístupní JavaScriptu štyri nové funkcie dokumentu: `execCommand()`, `queryCommandEnabled()`, `queryCommandState()`, `queryCommandValue()`. Najdôležitejšou z týchto funkcií je `execCommand()`, ktorá formátuje text, vkladá do dokumentu obrázky a odkazy, vracia zmeny dokumentu (undo/redo), atď. Funkcia `execCommand()` očakáva 3 parametre:

```
document.execCommand(command, userInterface, value)
```

Argumenty funkcie `execCommand()`:

`command`

Názov operácie, ktorú chceme vykonať. Niektoré operácie pracujú s označeným textom, niektoré s pozíciou textového kurzora a niektoré s celým dokumentom. Pozrite si tabuľku najdôležitejších operácií.

`userInterface`

Musí byť vždy *false*, inak funkcia vyhodí výnimku `NS_ERROR_NOT_IMPLEMENTED`.

`value`

Niektoré operácie vyžadujú dodatočný parameter. Napríklad ak vytvárame hypertextovy odkaz, ako parameter odovzdáme adresu, na ktorú má odkaz smerovať:

```
document.execCommand('createlink', false, 'http://www.abclinuxu.cz');
```

V praxi vykonáva funkcia `execCommand()` celú prácu za nás a nemusíme sa starať o nič iné, ako volanie tejto funkcie so správnymi argumentami. Rozšírme teda primitívny editor z minulej časti o niekoľko formátovacích funkcií:

```
<html>
<head>
  <title>Moj druhy WYSIWYG editor</title>
  <script type="text/javascript">
  var editor;
  function setup() {
    editor = document.getElementById('editor');
    editor.contentWindow.document.designMode='on';
  }
  function editor_alert() {
    alert(editor.contentWindow.document.body.innerHTML);
  }
  function editor_execCommand(command, value) {
    editor.contentWindow.document.execCommand(command, false, value);
  }
  </script>
```

```
</head>
<body onload="setup();">
  <p>Tento text nie je editovatelny... editujte obsah v ramci, prosim.</p>
  <iframe id="editor" src="template.html"></iframe>
  <p>Nastrojovy panel:</p>
  <ul>
  <li><a href="javascript: editor_alert();">
        Ukaz mi to HTML!</a></li>
  <li><a href="javascript: editor_execCommand('undo');">
        Undo</a></li>
  <li><a href="javascript: editor_execCommand('redo');">
        Redo</a></li>
  <li><a href="javascript: editor_execCommand('bold');">
        Tucne</a></li>
  <li><a href="javascript: editor_execCommand('italic');">
        Kurziva</a></li>
  <li><a href="javascript: editor_execCommand('underline');">
      Podskrtnute</a></li>
  <li><a href="javascript: editor_execCommand('strikethrough');">
      Preskrtnute</a></li>
  <li><a href="javascript: editor_execCommand('removeformat');">
      Normalne</a></li>
  <li><a href="javascript: editor_execCommand('formatblock', '<h1>');">
      Nadpis 1</a></li>
  <li><a href="javascript: editor_execCommand('formatblock', '<h2>');">
      Nadpis 2</a></li>
  <li><a href="javascript: editor_execCommand('formatblock', '<h3>');">
      Nadpis 3</a></li>
  <li><a href="javascript: editor_execCommand('formatblock', '<p>');">
      Odstavec</a></li>
  </ul>
</body>
</html>
```

Pridali sme do hlavného dokumentu ďalšiu JavaScriptovú funkciu `editor_execCommand()`. Očakáva 2 argumenty, ale pri volaní tejto funkcie si môžeme dovoliť druhý argument vynechať. V JavaScripte to nie je chyba, v tele funkcie bude tento argument nastavený na hodnotu `undefined` a táto hodnota je takmer to isté ako hodnota `null`. Funkcia `editor_execCommand()` jednoducho volá funkciu `execCommand()` editovaného dokumentu a odovzdá mu svoje vlastné dva argumenty. Pridávať môžeme aj ďalšie funkcie z nasledujúcej tabuľky:

Operácia	Hodnota	Popis
undo		Vráti jednu zmenu v dokumente.
redo		Znovu vykoná jednu vrátenú zmenu v dokumente.
formatblock	<h1>, <h2>, ..., <p>, <pre>, ...	Uzavrie odstavec textu do zvoleného HTML tagu.
insertimage	*adresa obrázku*	Vloží na dokument obrázok.
removeformat		Odstráni všetky formátovacie HTML značky z označeného textu.
createlink	*adresa*	Vytvorí z označeného textu hypertextový odkaz zo zvolenou adresou.
unlink		Zmení hypertextový odkaz na text.

Kompletný zoznam príkazov podporovaných editorom Midas nájdete na v špecifikácii Midasu [1]. Príkazy MSHTML Editoru sa nachádzajú v knižnici MSDN – Command Identifiers [2].

Odosielanie dokumentu na server

Odoslať HTML dokument po upravení na server môžeme viacerými spôsobmi. Najjednoduchšie je dokument zakódovať do adresy dopytovanej webovej služby ako argument. To znamená, že ak sme umiestnili webovú službu, ktorá dokument prijme a spracuje na adresu `http://example.com/service` a dokument vyzerá takto:

```
<h1>Ahoj svet!</h1><p>K&R napísali prvý
unix?</p>
```

tak pomocou JavaScriptu otvoríme v prehliadači adresu (jeden dlouhý řetězec, pozn. ed.)

```
http://example.com/service?text=%3Ch1%3EAhoj%20svet!%3C%2Fh1%3E%3Cp%3EK%\\
26R%20nap%C3%ADsali%20prv%C3%BDunix%3F%3C%2Fp%3E
```

Funkcia v JavaScripte

```
function send_document(host) {
window.location.href = host + '?html='
  + encodeURIComponent(editor.contentWindow.document.body.innerHTML);
}
```

Položka nástrojového panelu

```
<li><a href="javascript: send_document('http://example.com/');">
  Odoslat na example.com</a></li>
```

Záver

Nabudúce si prejdeme ďalšie možnosti odosielania dokumentu na server. Ďalej si ukážeme trik, ktorým uložíme dokument z Firefoxu na disk. Nakoniec si povieme pár slov o ostatných prehliadačoch podporujúcich WYSIWYG editor.

Odkazy

[1] http://www.mozilla.org/editor/midas-spec.html
[2] http://msdn.microsoft.com/workshop/author/dhtml/reference/commandids.asp

⋆⋆⋆

Jaderné noviny – 19. 7. 2006

Robert Krátký

Aktuální verze jádra: 2.6.17.6. Linux Kernel Summit 2006. Proč není Reiser4 v jádře.

Aktuální verze jádra: 2.6.17.6

Aktuální verze jádra je 2.6.17.6, vydaná [1] 15. července. Opravuje několik problémů způsobených verzí 2.6.17.5 [2], která obsahovala opravu lokální zranitelnosti umožňující získat práva roota v kódu souborového systému `/proc`. 2.6.16.25 [3] a 2.6.16.26 [4] byly vydány se stejnými opravami.

Aktuální předverze je 2.6.18-rc2, vydaná [5] 15. července. Obsahuje velké množství oprav a sadu patchů pro počítání doby, po kterou úlohy čekají [6]. Vizte podrobnosti v dlouhém changelogu [7]. Po vydání -rc2 bylo přidávání nových patchů do hlavní větve pozastaveno kvůli jadernému summitu a ottavskému linuxovému symposiu. Aktuální -mm strom je 2.6.18-rc1-mm1 [8]. Mezi nedávné změny patří podpora architektury Atmel a spousta oprav.

Linux Kernel Summit 2006

Linux Kernel Summit 2006 se tradičně konal dva dny před zahájením Ottawa Linux Symposium. Redaktor LWN Jonathan Corbet, člen programové komise summitu, tam byl a dělal si poznámky.

Den 1. – 17. července

První den jaderného summitu se diskutovalo o následujících věcech:

- Procesory – tři výrobci debatovali s vývojáři jádra o budoucích produktových plánech.
- Shrnutí mini-summitů [9] – poznámky z průběhu mini-summitů o ukládání dat, bezdrátovém síťování, souborových systémech, správě paměti a správě napájení.
- Kvalita jádra a vývojový proces [10] – Andrew Morton mluvil o tom, jestli má jádro skutečně problém s kvalitou, a jakými způsoby by šlo vylepšit vývojový model.
- Rozhraní `ioctl()` [11] – věnovalo se tvrzení, že toto často kritizované systémové volání není vždy jen špatné.
- Jaderné ABI [12] – jak se vyhnout změnám a jak nejlépe spravovat nástroje úzce vázané na jádro.
- Software suspend [13] – co bude třeba, aby fungovalo spolehlivě; je dobrý nápad dávat software suspend do uživatelského prostoru?
- Dokumentace [14] – aktuální stav a co lze udělat pro jeho zlepšení.

Den 2. – 18. července

Druhý a poslední den jaderného summitu měl na programu následující setkání:

- Realtime – stav realtime patchů.
- Embedded systémy [15] – a co je potřeba k jejich lepší podpoře.
- Bezpečnost [16] – konkrétně osud AppArmor a systému Linux Security Modules (LSM – linuxové bezpečnostní moduly).
- Paravirtualizace a kontejnery [17] – a jak zprovoznit virtualizační řešení v jediném hlavním jádře.
- Automatizované testování [18] – snaha o zachycení chyb v jádře předtím, než přijde k uživatelům.
- Vrstva VFS.
- Škálovatelnost [19] – jak moc může Linux růst?

- DMA a IOMMU [20].
- Vývojový proces II [21] – poslední pohled na proces vývoje a uzavření summitu.

Shrnutí: dle mého názoru šlo o jeden z úspěšnějších jaderných summitů. Diskuze byly živé a zajímavé, témata relevantní. Došlo k několika rozhodnutím. Ačkoliv je vždy co zlepšovat, vypadá to, že vývojový model funguje dobře a vývojáři spolu povětšinou spolupracují. Věci se pohybují poměrně hladce kupředu, takže summit byl také úspěšný.

Následující obsah je ©KernelTrap

Proč není Reiser4 v jádře

17. črc, originál [22]

Otázka zda a kdy bude Reiser4 začleněn do hlavního jádra, je probírána už dva roky. Hans Reiser souborový systém popsal v březnu 2004 jako poměrně stabilní pro průměrné uživatele.

Byl zařazen do -mm stromu Andrew Mortona, ačkoliv problémy s Reiser4 pluginy a stylem psaní kódu způsobily před dvěma lety dlouhé diskuze. Dvě nedávná vlákna na ⋄LKML [23] otázku znovu otevřela. Na obecné, netechnické úrovni se ptala, proč ještě nebyl Reiser4 začleněn do hlavního jádra. Někteří se nechali slyšet, že jde o politické rozhodnutí, jiní uváděli jako důvod obavy, že Namesys by po začlenění mohla na Reiser4 zapomenout a začít se zabývat jiným souborovým systémem. Odpovědi na tyto teorie ukazují, že ve skutečnosti existují technické překážky, které je potřeba před začleněním vyřešit, a že bylo v tomto ohledu odvedeno hodně práce. Další diskuze lze najít na nedávno vytvořené stránce v rámci kernel newbies wiki [24]. Hans Reiser poslal seznam úkolů pro nejbližší období,

které by měly dát do pořádku zbývající technické otázky. Seznam obsahuje začlenění batch_write a kódu optimalizace čtení do -mm, dokumentace všeho ve wiki Namesys, zjištění a vyřešení hlášených zatuhnutí systému při používání Reiser4, kompletní kontrolu kódu crypt-compress, optimalizaci fsync, kontrolu instalačních instrukcí a kontrolu jaderné dokumentace. Hans vysvětlil: Stabilita našeho kódu bohužel trochu utrpí kvůli všem těmto změnám – to nelze vyřešit jinak než časem. Na druhou stranu teď používáme méně procesoru. Jedinou výkonnostní slabinou Reiser4 je nyní fsync. Jakmile bude připraven kód crypt-compress, vydáme Reiser4.1-beta (protože máme pluginy, znamená pro uživatele vydání betaverze to, že když připojí oddíl pomocí mount -o reiser4.1-beta, bude cryptcompress výchozím pluginem; pokud ne, budou dále používat Reiser4.0). Zdvojnásobení výkonu a snížení využití disku o polovinu bude legrace.

Odkazy

[1] http://lwn.net/Articles/191512/
[2] http://lwn.net/Articles/191488/
[3] http://lwn.net/Articles/191487/
[4] http://lwn.net/Articles/191511/
[5] http://lwn.net/Articles/191513/
[6] http://lwn.net/Articles/173882/
[7] http://kernel.org/pub/linux/kernel/v2.6/testing/ChangeLog-2.6.18-rc2
[8] http://lwn.net/Articles/191375/
[9] http://lwn.net/Articles/191652/
[10] http://lwn.net/Articles/191650/
[11] http://lwn.net/Articles/191653/
[12] http://lwn.net/Articles/191654/
[13] http://lwn.net/Articles/191657/
[14] http://lwn.net/Articles/191659/
[15] http://lwn.net/Articles/191822/
[16] http://lwn.net/Articles/191737/
[17] http://lwn.net/Articles/191923/
[18] http://lwn.net/Articles/191924/
[19] http://lwn.net/Articles/191929/
[20] http://lwn.net/Articles/191931/
[21] http://lwn.net/Articles/191932/
[22] http://kerneltrap.org/node/6844
[23] http://www.abclinuxu.cz/slovnik/lkml
[24] http://wiki.kernelnewbies.org/WhyReiser4IsNotIn

⋆⋆⋆

Jaderné noviny – 26. 7. 2006

Robert Krátký

Aktuální verze jádra: 2.6.17.7. Citát týdne: Hans Reiser. Nový pohled na síťové kanály. revoke() a frevoke(). Souborové systémy, politika a jádro. Linus o Extensible Firmware Interface.

Aktuální verze jádra: 2.6.17.7

Aktuální verze jádra řady 2.6 je 2.6.17.7, vydaná [1] 24. července. Doplňuje poměrně dlouhou řadu oprav pro problémy se síťováním, zvukem a několika dalšími oblastmi. *Aktuální předverze je i nadále 2.6.18-rc2*. V hlavním git repozitáři se hromadí opravy a brzy lze očekávat vydání -rc3. Od 2.6.18-rc1-mm2 [2] ze 14. července nevyšla žádná nová verze stromu -mm.

Citát týdne: Hans Reiser

Problémem Linuxu je, že jeho úspěch k němu přivádí lidi schopnější než ty, se kterými se začínalo, a nedaří se mu s tím vypořádat. Je v tom přímo mizerný, protože se chová nejhorším možným způsobem. Ztratili jsme docela dost vývojářů souborových systémů, protože se prostě nechtěli dohadovat s lidmi, kteří toho ví méně než oni, ale na nové návrhy se tváří otráveně a chovají se nezdvořile, protože chtějí být bossové.

– Hans Reiser [3]

Nový pohled na síťové kanály

Když Van Jacobson nadhodil [4] v lednu minulého roku na linux.conf.au svou představu o síťových kanálech, způsobil v linuxové síťové komunitě pořádný rozruch. Pomocí výrazných změn způsobu zpracovávání příchozích paketů a natlačením většiny práce co nejblíže k cílové aplikaci dosáhl Van velkého zlepšení výkonu – odstranil až 80 % režie na víceprocesorových systémech. S takovými čísly to vypadalo, že o začlenění kanálů do Linuxu je už dopředu rozhodnuto. Od té doby se však začala o slovo hlásit realita – to ona dělává, dříve či později. Proto také David Miller posledně [5] říkal o síťových kanálech tohle:

Na síťové kanály od VJ nebuďte příliš natěšení, protože se objevuje stále více a více překážek bránících jejich nasazení v praxi... Všechen výkon původně ušetřený na úkor routování, netfilteru a vyhledávání TCP soketu se nám ztrácí, když se snažíme přimět síťové kanály k fungování na skutečných systémech.

Daný problém se týká integrace síťových kanálů a netfilteru. Doufalo se, že by pakety mohly být identifikovány a roztříděny do příslušných kanálů ještě před zpracováváním netfilterem (firewallem). Pak by to zpracovávání mohlo být provedeno blízko aplikaci, na tom samém procesoru. Ukázalo se však, že netfilter může změnit skutečný cíl paketu. Takže je nutné pakety filtrovat před vstupem do kanálu a většina výkonu získaného díky použití kanálu je ztracena. Alexej Kuzněcov poslal podrobnou kritiku kanálů [6], ve které tvrdí, že většina výhod je iluzorních:

Je to úžasná hračka. Ale nevidím nic, co by ji mohlo pozvednout na praktickou úroveň. Exokernely tohle dělaly léta a veškerý získaný výkon je vyrovnán příliš komplikovaným klasifikačním enginem, který musí zůstat v jádře a dělat v podstatě totéž, co dělá routování/firewall/...

Kromě toho se zdá, že mnoha výhod, které nabízejí kanály, lze dosáhnout pečlivým využitím možností moderního hardwaru. Konkrétně jde o to, že stále větší počet zařízení dokáže provádět jednoduchou klasifikaci paketů a směrovat je na procesor (přes cílená přerušení), kde běží cílová aplikace. Tato technika eliminuje přehmaty keše způsobované zpracováváním přerušení na jednom procesoru a protokolu na druhém. Vypadá to, že další zdánlivě chytrý nápad se nedostane do reálného nasazení. Některé z jeho základních konceptů, jako například datové struktury spolupracující s kešemi však pravděpodobně ovlivní budoucí směr síťového

stacku. Takže ačkoliv asi nebude revoluční nový mechanismus, některá slibovaná zlepšení výkonu by přeci jen měla být realizována. A jak říká David [7]: Aspoň není potřeba psát tolik kódu.

revoke() a frevoke()

V některých variantách Unixu je systémové volání `revoke()`:

```
int revoke(const char *path);
```

Toto volání slouží k odpojení procesů od souborů; když je zavoláno s určitou `path`, uzavře všechny otevřené popisovače souborů, které odkazují na soubor nacházející se na konci této cesty [path]. Původním účelem bylo zbavit se lidí, kteří napsali program sedící na sériovém portu a předstírající, že je `login`. Jakmile bylo `revoke()` zavoláno se souborem zařízení odpovídajícím sériovému portu, všechny falešné loginy byly odpojeny a nemohly nikoho oklamat. Existují i jiná možná využití. Například odpojení procesu od souboru, který brání odpojení souborového systému. Linux toto systémové volání nikdy neměl, ale možná nebude trvat dlouho a dojde ke změně; Pekka Enberg poslal implementaci `revoke()` [8] ke kontrole. Pekka také přidal druhou verzi:

```
int frevoke(int fd);
```

Tato verze bere samozřejmě jako parametr otevřený popisovač souboru. V obou případech musí volající proces buď soubor vlastnit, nebo být schopen přebít práva. Takže `revoke()` umožňuje procesu vyškubnout otevřený soubor procesům, které vlastní jiní uživatelé – za předpokladu, že proces daný soubor vlastní. Zvládnout tuto operaci správně není tak docela snadné, což má za následek určité kompromisy v současné implementaci, které se možná nebudou některým vývojářům líbit. Zjednodušeně ten proces vypadá takto:

- Kód prochází ve smyčce přes každý proces v systému; u každého procesu projde tabulkou otevřených souborů a hledá popisovače souborů odpovídající zavíranému souboru. Pokaždé, když nějaký najde, vynuluje záznam popisovače (čímž se popisovač stane pro původního majitele nedostupným). Soubor však není uzavřen; místo toho je vytvořen seznam souborů pro pozdější akci.

  To vše je poměrně pomalé, ale problém to není: `revoke()` není operace, která by ovlivňovala výkon. Trochu problematičtější je alokace paměti (pro přidání záznamu do seznamu souborů k uzavření); selže-li, ukončí se `revoke()` na půli cesty, přičemž provedlo neznámé množství škod, aniž by splnilo svůj účel.
- Po uzavření všech otevřených popisovačů souborů mohou být uzavřeny samotné soubory. `revoke()` projde vytvořeným seznamem a všechny otevřené soubory uzavře.
- Zůstává jeden malý nepříjemný problém: některé procesy možná využily `mmap()` k namapování souboru do svých adresných prostorů. Volání `revoke()` musí s těmito oblastmi paměti něco udělat, nebo jeho práce nebude dokončena. Takže je nutné projít všechny oblasti virtuální paměti spojené se souborem; u každé je metoda `nopage()` nastavena na speciální verzi, která vrátí chybový stav.

  Tato změna zabrání procesu v chybném běhu na nových stránkách z uzavřeného souboru, ale neudělá nic se stránkami, které už jsou součástí adresného prostoru procesu. Pro nápravu je potřeba prolézt tabulky stránek každého procesu, který soubor namapoval, a vyčistit všechny záznamy, které odkazují na stránky z daného souboru.

Alternativní přístup používá patch pro nucené odpojení [9] od Tigrana Aivaziana, který byl během své dost dlouhé historie (komentáře k němu obsahují jméno autora portu na jádro 2.6) upravován i mnoha jinými vývojáři. Patch má jiný cíl – být schopen odpojit souborový systém bez ohledu na aktuální činnosti. Musí však vyřešit stejný problém – uzavření přístupu ke všem souborům na cílovém souborovém systému. Místo vyčištění popisovačů souborů nahrazuje tento patch výchozí strukturu `file` novou, která pochází ze souborového systému "badfs". Po této změně budou na pokusy o práci se souborem vraceny `EIO`. Mapování paměti je odstraněno přímým voláním `munmap()`.

Konečná podoba patche by klidně mohla být kombinací obou a poskytovala by jak nucené odpojení, tak `revoke()`. Do té doby bude nutné vyřešit několik zbývajících otázek (například jak provést bezpečné

uzamčení bez zpomalení vysoce optimalizovaných `read()` a `write()`). O tyto funkce je však zjevně zájem, takže je pravděpodobné, že práce dospěje až k začlenění do jádra.

Následující obsah je ©KernelTrap

Souborové systémy, politika a jádro

24. črc, originál [10]

Na LKML pokračuje diskuze o tom, proč nebyl Reiser4 dosud začleněn do jádra. Hans Reiser postavil do kontrastu snahy o začlenění Reiser4 a nedávnou diskuzi o novém souborovém systému ext4. Ten kód není ještě ani napsán nebo testován, ale do jádra jde rovnou, aby se jeho vývojáři nemuseli starat o udržování patchů mimo hlavní strom. No ne. Tak trochu těžko vyvracet, že nejde o politikaření, co? Theodore T'so odpověděl: Jde o vývojovou proceduru, které bylo dosaženo po diskuzi a hledání kompromisu na LKML a mezi vývojáři ext2/3/4. Není to původní plán, který navrhovali vývojáři ext2, ale když jsme naslouchali obavám a návrhům, nepochybovali jsme o motivech těch lidí, kteří s návrhy přicházeli; naslouchali jsme.

Pokračoval poznámkou o tom, že části kódu, ze kterého bude ext4, byly napsány už před rokem, a byly důkladně testovány a kontrolovány. Další připomněli, že evoluce z ext3 na ext4 bude velmi veřejný proces, přičemž patche budou začleňovány postupně, kdežto Reiser4 je založen na úplně jiném kódu než Reiser3.

Linus o Extensible Firmware Interface

26. črc, originál [11] Nedávný patch poslaný do LKML obsahoval následující popis: Tento patch přidává mapování paměti efi na e820.

Andrew Morton reagoval: Proč?

Linus Torvalds nabídl svůj pohled na EFI (Extensible Firmware Interface – rozšiřitelné firmwarové rozhraní), který uvedl slovy: Další dementní věc od Intelu (první bylo ACPI).

EFI je náhradou ⋄BIOSu [12], která byla původně navržena pro použití na architektuře ia64, ale od té doby ji přijaly i jiné počítače, včetně intelských Maců. Linus pokračoval ve vysvětlování: Původní EFI kód v jádře v podstatě duplikuje všechna rozhraní BIOSu (tj. vše, co se dotýká mapování paměti, máme ve dvou variantách: normální, otestovaná verze pro e820 BIOS a většinou nefunkční, splácaná verze EFI). Překládat paměťovou mapu EFI do e820 je tedy velmi rozumné. Linus pokračoval v dalším emailu: Abych se vyjádřil jasně – problém s EFI je ten, že na povrchu vypadá o hodně lépe než BIOS, ale v praxi se z toho stává jedna z těch věcí, které mají jen pár skutečných výhod a často jen přidávají spoustu komplikovanosti, protože ta 'nová a vylepšená' rozhraní byla z většiny vymyšlena komisí.

A pokračoval: Takže EFI má bezvadný shell, systém pro natahování ovladačů a další pěkné funkce. Kde „pěkné" samozřejmě znamená „daleko komplikovanější než v sedmdesátých letech, když byli lidi ještě blbí a jen chtěli, aby věci fungovaly". Je samozřejmě otázkou, jestli lidi za posledních 30 let zmoudřeli nebo zblbli. Není to dost času na to, abychom měli díky evoluci větší mozky, ale určitě to stačilo na to, aby většina lidí přestala rozumět tomu, jak vlastně hardware funguje.

A co se BIOSu týče: Nikdy bych netvrdil, že BIOS je úžasný, ale každý aspoň ví, že je to jen zavaděč systému, a nikdo se z něj nesnaží udělat něco jiného.

Odkazy

[1] http://lwn.net/Articles/192695/
[2] http://lwn.net/Articles/191375/
[3] http://lwn.net/Articles/192762/
[4] http://lwn.net/Articles/169961/
[5] http://lwn.net/Articles/192769/
[6] http://lwn.net/Articles/192772/
[7] http://lwn.net/Articles/192774/
[8] http://lwn.net/Articles/192107/
[9] http://developer.osdl.org/dev/fumount/kernel2/patches/2.6.13-rc3/2/
[10] http://kerneltrap.org/node/6876
[11] http://kerneltrap.org/node/6884
[12] http://www.abclinuxu.cz/slovnik/bios

⋆ ⋆ ⋆

Jaderné noviny – 2. 8. 2006

Robert Krátký

Aktuální verze jádra: 2.6.17.8. Citát týdne: Linus Torvalds. Marcelo Tosatti předává štafetu jádra 2.4. Filtrování SCSI příkazů. Diskuze o Reiser4 – znovu. Rozhraní pro jaderné události. Nová jádra a staré distribuce.

Aktuální verze jádra: 2.6.17.8

Aktuální předverze je 2.6.18-rc3, vydaná [1] 29. července. Přísun patchů se zpomaluje úměrně s tím, jak začíná být nová verze stabilní. Takže tato předverze přidává několik oprav, ale nic moc navíc. Podrobnosti jsou v dlouhém changelogu [2]. Od vydání -rc3 bylo do hlavního repozitáře začleněno přes 100 oprav. Aktuální -mm strom je 2.6.18-rc2-mm1 [3]. Mezi nedávné změny patří velká aktualizace x86-64 a NFS a spousta oprav.

Citát týdne: Linus Torvalds

Tvrdím, že rozdíl mezi špatným a dobrým programátorem je v tom, jestli považuje za důležitější svůj kód nebo datové struktury. Špatní programátoři se starají o kód. Dobří programátoři se starají o datové struktury a jejich vztahy.

– Linus Torvalds [4]

Marcelo Tosatti předává štafetu jádra 2.4

Marcelo Tosatti oznámil [5] vydání jádra 2.4.33-rc3, které obsahuje jen pár patchů. Zároveň dal na vědomí, že správcovství jaderné řady 2.4 přebírá Willy Tarreau, který již nějakou dobu připravoval sérii patchů "hotfix" pro 2.4. Marcelovi, který řadu 2.4 spravoval od verze 2.4.16, patří velký dík.

Filtrování SCSI příkazů

Pálení dat na CD nebo DVD je komplikovaný proces, při němž se používá velké množství SCSI příkazů. Takže každá aplikace, která vypaluje disky, musí mít schopnost posílat mechanice speciální SCSI příkazy. Těsně před vydáním jádra 2.6.8 se však vývojáři rozhodli, že aplikacím by nemělo být dovoleno posílat *jakékoliv* SCSI příkazy. Některé z těchto příkazů by mohly vést k přepsání firmwaru mechaniky, vznícení nebo nahrazení hudebních stop nahrávkami zpěvu Richarda Stallmana. Ve snaze zabránit těmto nežádoucím věcem začlenil Linus na poslední chvíli patch [6], který uživatelům bez dostatečných práv zakazoval posílat SCSI příkazy, jež nebyly na seznamu povolených v jádře. Je skoro jisté, že žádný uživatel nezničil CD mechaniku jádrem 2.6.8. Jen málo z nich vůbec mohlo zapisovat na disky; filtrování bylo tak přísné, že uživatelé bez práv nemohli dělat nic užitečného. Pozdější aktualizace to napravily a s 2.6.10 už zase vypalování většině uživatelů fungovalo.

Ne však všem uživatelům. Jak nedávno poznamenal [7] Dave Jones v konferenci linux-scsi, filtrování příkazů pořád působí potíže některým mechanikám Plextor. Utilita cdrecord se pokouší posílat těmto mechanikám speciální příkazy daného výrobce, ale jádro je nepustí. Všechno se zastaví a uživatel musí práci začít znovu jako root. Dave se ptal: nebylo by vhodné přidat k filtrovacímu kódu možnost výjimek pro příkazy specifické u různých výrobců?

Vývojáři blokového subsystému reagovali tak, že by se mělo prostě vyhodit celé filtrování příkazů. Tato funkce byla projednávána na nedávném setkání (storage summit) a účastníci se shodli, že by měla být odstraněna. James Bottomley shrnul [8]:

Dovolíme-li uživatelům vypalovat CD, není možné je kontrolovat bezchybně, jak ukazuje tento případ. A když povolíme příkazy výrobců, určitě se najdou takové, které zformátují disk nebo zničí firmware... Takže myslím, že je lepší vyhodit tu tabulku a přiznat, že nenabízíme žádné zabezpečení, než předstírat, že nějaké máme.

Vůči filtrovacímu kódu je několik výhrad. Jde o zápis pravidel do jádra, což všeobecně nebývá kladně přijímáno – i když v tomto případě jde vlastně o pokus rozlišit přístup k disku v mechanice a k mechanice samotné. Seznam příkazů nikdy nebude dokonalý. Vypadá to, že některé mechaniky potřebují slyšet speciální zaklínadla od výrobce, jinak odmítnou na disk zapisovat. Některé příkazy mají na různých mechanikách různé významy. Co je bezpečné pro vypalovačku CD, může být likvidační pro jiný typ SCSI zařízení. Nepomáhá ani to, že v jádře jsou ve skutečnosti dva různé filtry SCSI příkazů (jeden v `drivers/scsi/sg.c`, druhý v `block/scsi_ioctl.c`), které implementují různá pravidla. Všechny tyto důvody pravděpodobně přispěly k rozhodnutí účastníků setkání – odstranit filtrování. Ten plán má jedinou malou chybku: Linus má na filtrování jiný názor [9]:

Jinými slovy: filtrování příkazů z block/scsi_ioctl.c odstraníte jedině v jádře, které nespravuji já, nebo ho zrušíte jiným způsobem, který bude tak maskovaný, že si toho nevšimnu. Protože nejsem tak hloupý, abych si myslel, že je v pořádku nechat běžné uživatele nastavovat hesla ovladače nebo přepisovat firmware disku jen proto, že mají na zařízení právo zápisu. Berte to jako mé poslední slovo.

Takové prohlášení vypadá dost definitivně. Což však neznamená, že situaci nelze zlepšit. V současné době se prosazuje myšlenka nechat oprávněného uživatele provádět v tabulce pro filtrování příkazů změny. Distribuce by pak mohly dodávat nástroje, které rozpoznají problematická zařízení a podle toho upraví filtrovací tabulky. Celé by to mohlo být transparentně integrováno s funkcí hotplug. Jens Axboe napsal patch (původně byl autorem Peter Jones), který promění filtrovací seznam v samostatný objekt pro každé zařízení. Ten pak půjde nastavit prostřednictvím sysfs, takže každé zařízení by mohlo mít vlastní sadu výjimek. Jak přesně by takové rozhraní pracovalo, to je třeba ještě prodiskutovat. Ale konfigurovatelné filtry pro každé zařízení zvlášť se zdají jako cesta správným směrem. Zůstává zachováno filtrování nebezpečných příkazů, ale rozhodování o pravidlech se přesunuje do uživatelského prostoru. Jakmile bude možné pravidla měnit, postarají se distributoři o to, aby byla jednotlivá zařízení dobře podporována. Nebo, pokud budou chtít, označí všechny příkazy jako povolené, a tím odstraní filtrování úplně.

Diskuze o Reiser4 – znovu

Jestli Hansi Reiserovi nějaká vlastnost chybí, není to vytrvalost. V říjnu 2002 požádal [10] o začlenění svého nového souborového systému reiser4 do vývojového jádra 2.5 předtím, než přejde do stabilizační fáze před vydáním 2.6. Uplynuly téměř čtyři roky, během kterých prošel reiser4 nekonečnými debatami v konferenci ◇linux-kernel [11], mnoha změnami opravujícími nalezené problémy, odstraněním základních funkcí a dlouhým čekáním v jádře -mm. Přes to všechno stále reiser4 není v hlavním jádře – ale Hans se nevzdává.

Do této chvíle bylo nutno překonat množství překážek. Funkce „soubory jako adresáře“ upravovala POSIXovou sémantiku způsobem, který některým lidem nebyl milý, a navíc způsobovala zásadní potíže se zamykáním; funkce byla odstraněna. Ne všichni pozorovatelé považují výsledky výkonnostních testů za důvěryhodné. Přetrvávají obavy z toho, jak se budou vývojáři reiser4 o souborový systém starat, jakmile bude začleněn. Hans je známý svými problémovými vztahy s ostatními vývojáři jádra a nereaguje vždy zrovna uhlazeně na (ne vždy zrovna uhlazené) komentáře kritiků. Výsledkem je trnitá cesta k začlenění souborového systému, který však skutečně nabízí zajímavé nápady a potenciálně špičkový výkon.

Částečně kvůli pocitu, že se proces začleňování reiser4 táhne už příliš dlouho, se do konference debata vrátila. Hans a spol. by rádi dosáhli zařazení reiser4 do jádra 2.6.19 a vypadá to, že by nakonec mohli uspět. Stále je tu několik otázek, i když některé z nich možná nejsou tak problematické, jak si občas lidi myslí. Největší z nich je pravděpodobně koncept pluginů. Pluginy souborovému systému umožňují chovat se jinak ke každému uloženému souboru; je díky nim možné přidat funkce jako komprimaci nebo šifrování

a další tajemné věci, které se teď dělají pomocí FUSE. Pluginy však ve vývojářské komunitě spustily hodně varovných signálů. Například Linus říká [12]:

Dokud jim říkáte „pluginy“ a chováte se k nim tak, nemám (a tuším, že spousta dalších lidí také ne) o to vůbec zájem. A abych řekl pravdu, spousta lidí si pomyslí, že hlavním účelem je buď podkopat copyrightová pravidla jádra, nebo přinejmenším vytvořit zmatek nekompatibilních sémantik bez rozumných pravidel pro zamykání atd.

Jeff Garzik si také dělá starosti [13]:

Nechtěl bych zajišťovat technickou podporu pro distributora, za nímž přijde zákazník s padlým reiser4 v případě, kdy si zákazník tajně nahradil standardní inodovou strukturu dat interně napsaným pluginem a rovněž tajně nahradil adresářový algoritmus uzavřeným pluginem od VyberSiProdejce. Nechtěl bych se tím zmatkem probírat debuggerem. Vývojáři chtějí, aby tvůrci reiser4 vzali na vědomí fakt, že podobný mechanismus, má-li vůbec smysl, by měl být implementován v rámci vrstvy VFS, ne v určitém souborovém systému. Reiser4 pluginy jsou vnímány jako samostatný, soukromý VFS, který může nadělat spoustu potíží.

Co si však mnoho lidí neuvědomuje, je to, že otázka pluginů je teď daleko méně problémová než bývala. Nelze je natáhnout za běhu, takže není nutné mít starost kvůli copyrightu jako u uzavřených jaderných modulů. A většina funkcí pluginů byla na základě komentářů odstraněna. Andrew Morton, který kód nedávno prohlížel [14], poznamenal:

Vypadá to, že jsou pluginy označovány za příliš velkého strašáka – jde jen o interní abstrakční vrstvu, která umožňuje pozdější čisté a bezpečné přidávání funkcí. Určitě nestojí za všechen ten povyk.

Podle Andrewa je největším problémem absence přímého I/O a podpory rozšířených atributů. Přímý I/O možná není příliš vzdálen, ale nevypadá to, že by mohly být v blízké budoucnosti připraveny i rozšířené atributy. To mimo jiné znamená, že by reiser4 nemohl mít podporu SELinuxu. Takové omezení by mohlo způsobit, že někteří distributoři vynechají podporu reiser4, i kdyby byla začleněna v hlavním jádře.

Zbývající stížnosti by mohly stačit k tomu, aby odradily některé uživatele či distributory od používání reiser4, ale zdá se, že nejsou dostatečné k tomu, aby zabránily zařazení do hlavní větve. Nový souborový systém se netýká těch, kteří jej nepoužívají, a problémy, které měli jeho uživatelé (například zamrzání) jsou snad už dávno pryč. Takže Hansovo dlouhé čekání se možná chýlí ke konci.

Rozhraní pro jaderné události

Článek o síťových kanálech [15] z minulého týdnu naznačil, že kanály možná nejsou řešením budoucnosti. Od té doby proběhlo hodně diskuzí o tom, jak by se síťování mohlo pohnout kupředu – a kanály by mohly ještě hrát roli. Na rozdíl od jiných operačních systémů postrádá Linux systémové volání pro obecné hlášení o událostech. Místo toho používají linuxové aplikace volání jako `poll()`, aby zjistily, kdy čeká nějaká práce. `poll()` bohužel problém neřeší úplně, takže aplikační smyčky událostí musí dělat komplikované věci, aby se vypořádaly například se signály. Zpracovávání asynchronního I/O v rámci tradiční linuxové smyčky událostí může být obzvlášť nepříjemné. Kdyby existovalo jediné rozhraní, který by aplikaci poskytlo veškeré potřebné informace o událostech, byly by aplikace mnohem jednodušší. Je to také příležitost k výraznému zvýšení výkonu. Jsou dva návrhy linuxového rozhraní pro události: mechanismus kevent [16] a API kanálu událostí navrhované Ulrichech Drepperem [17] na letošním linuxovém symposiu v Ottavě. Ve výhodnější pozici je teď kevent, protože existuje funkční implementace, kterou si lze prohlédnout. Většina diskuze se tedy zaměřovala na to, jak kevent vylepšit. Původní API kevent je považováno za trochu moc složité; závisí na jediném multiplexním systémovém volání (`kevent_ctl()`), což je přístup, který není všeobecně moc schvalován. Volání také vyžaduje, aby aplikace sestavila pole se dvěma různými typy struktur, a to je poněkud neohrabané. Jedním z prvních návrhů tedy bylo oddělit jednotlivé části API. Aktuální kevent patch [18] (1. srpna) obsahuje nové systémové volání:

```
int kevent_get_events(int ctl_fd,
                      unsigned int min_nr,
```

```
unsigned int max_nr,
unsigned int timeout,
void *buf,
unsigned flags);
```

Toto volání by vrátilo mezi `min_nr` a `max_nr` událostmi a ukládalo by je postupně do `buf`, dokud by nevypršel `timeout` (určený v milisekundách). Parametr `flags` není v současné implementaci využit.

Na tomto rozhraní by šlo vylepšit hned několik věcí, ale jak se zdá, finální podoba bude pravděpodobně dost odlišná. Současné rozhraní i nadále vyžaduje častá systémová volání kvůli stahování událostí; linuxová systémová volání jsou sice rychlá, ale při velkém vytížení by stejně bylo lepší strávit pokud možno více času v uživatelském prostoru. Mohlo by to být možné při použití jiného přístupu k hlášení o událostech. Nápad, o kterém se mluvilo, navrhoval namapování pole `kevent` struktur mezi jádrem a uživatelským prostorem. S takovým polem by se zacházelo jako s kruhovým bufferem a mohlo by být spravováno pomocí indexového mechanismu podobného kanálům, který by příliš nezatěžoval keš. Jádro by do bufferu ukládalo události ve chvíli, kdy se staly, a uživatelský prostor by je zpracovával. Kdykoliv by byla událost ke zpracování, aplikace by ji získala bez jakéhokoliv zásahu do jádra. Jakmile by takový mechanismus fungoval, mohlo by být volání `kevent_get_events()` nahrazeno prostým rozhraním „počkej na události“ (ačkoliv glibc by téměř jistě nabídla synchronní funkci „získej události“). Výsledkem by bylo velmi rychlé rozhraní – zvláště při vysokém počtu událostí.

Stále je však potřeba vyřešit pár problémů. První se týká situace, kdy se buffer zaplní. Současné asynchronní I/O rozhraní neumožňuje, aby bylo více čekajících operací než je dostupných kontrolních blokových struktur; tak je zaručeno, že bude dost místa na hlášení o stavu každé operace. To může být důležité, protože ta část jádra, která chce provádět hlášení, často běží na úrovni softwarového nebo hardwarového přerušení. Pokud si však představíme použití kevents pro sledování tisíců otevřených soketů, neznámého počtu událostí týkajících se spojení atd., začne být čím dál více nepraktické alokovat všechny struktury událostí napřed. Při zaplnění bufferu se tedy bude muset provést něco chytrého.

Druhý problém se týká událostí „spuštěných úrovní“ [level-triggered], které více odpovídají specifickému stavu než skutečné události, která proběhla. „Na tento soket lze zapisovat“ je příkladem takové události. Když se pro zjišťování toho, jestli bude zápis blokován, používá rozhraní jako `poll()`, může jádro ověřit stav a okamžitě vrátit, je-li možné na daný popisovač souboru zapisovat. Hlášení takových věcí přes kruhový buffer je o dost těžší. Ať tak nebo tak, aplikace budou muset tento typ událostí zjišťovat přímo. Vzhledem k tomu, že se jedná o exponované téma, je pravděpodobné, že se brzy objeví způsob, jak tyto potíže vyřešit. To by mohlo zamést cestu pro začlenění kevents do hlavního jádra třeba už v 2.6.20.

Nová jádra a staré distribuce

Utilita `udev` má jasně určený úkol: brát informace z jaderných událostí a virtuálního souborového systému sysfs a používat je k vytváření souborů zařízení odpovídajících skutečné konfiguraci systému. Když `udev` selže, je systém částečně nebo zcela nepoužitelný – situace, která se uživatelům obvykle dost nelíbí. Takže když Andrew James Wade nahlásil [19] selhání `udev` při použití nedávného -mm jádra, vývojáři zpozorněli.

Ukázalo se, že problém je způsoben určitými změnami v sysfs, které mají zlepšit správu napájení. Konkrétní chybu lze opravit přidáním jiného patche, ale ten na oplátku vede k dalším potížím; několik distrucí přestává fungovat, protože verze `udev`, kterou dodávají, je příliš stará na to, aby rozuměla novému formátu sysfs. Andrew Morton si stěžoval [20], že Fedora Core 3 nefunguje, ale problém bude pravděpodobně rozšířenější. Greg Kroah-Hartman, vývojář zodpovědný za změny, reagoval [21] takto:

To distro už není podporované, že jo? Jak dlouho bys chtěl, aby jádro podporovalo nepodporovaná komunitní distra, která jinak těží z faktu, že jsou rychle aktualizována? [...] A ano, stáhnu z hlavního jádra ten patch, který lidi nutí upgradovat na udev z FC5, a počkám s ním na další verzi (minumum pak bude 081, která byla vydána v lednu 2006 – až bude venku 2.6.19, bude už 10 měsíců stará).

Andrew nebyl nadšen [22]:

Jde mi (opět) o to, že se bavíme o zmrzačení *všech* dister starších deseti měsíců. Nehrajeme si přeci na „ahá, ale to už nepodporujeme“... Fakt blbé. Víš, jaké všechny stroje přestanou fungovat? Já tedy ne.

Mezi distribuce, které by s vydáním 2.6.19 přestaly fungovat, patří mimo jiné i Ubuntu 6.06 LTS („dapper“) a ještě nevydaný Slackware 11. Takže není překvapivé, že se takové změny nelíbily více lidem. Je dost pravděpodobné, že bude celá sada patchů stažena a znovu promyšlena. Greg však pokládá [23] zásadní dotaz:

Jak dlouho by se měla komunita starat o distro po té, co jej opustili jeho tvůrci?

Tradiční odpověď by zněla „navždy“, ale s novou generací nástrojů, které přenášejí „jádro do uživatelského prostoru“, je těžké takový slib dodržet. Nástroje jako `udev` jsou těsně propojené se souborovým systém sysfs, který je zase téměř přímou reprezentací interních jaderných datových struktur. Sysfs svým způsobem funguje jako interní jaderné API, ale ve skutečnosti jde o rozhraní pro uživatelský prostor. Udržovat jej stabilní a vyhýbat se problémům s nekompatibilitou se staršími uživatelskými nástroji dá hodně práce. Navíc to ztěžuje skutečnost, že se vývojáři jádra pořád snaží vymyslet, jak by měl sysfs vlastně fungovat.

Na letošním jaderném summitu [24] se mluvilo o přibalení nástrojů jako `udev` ke zdrojovým kódům jádra a distribuování celého balíku dohromady. Nová jádra by byla vždy doprovázena novými funkčními verzemi `udev` a některé z problémů s nekompatibilitou by zmizely. Není však možné takto přibalit neomezené množství nástrojů a každopádně je těžké vnímat takový přístup jako cokoliv jiného než hack pro obejití skutečného problému, který představuje udržování stability tak širokého a komplexního ABI.

Tento konkrétní problém bude pravděpodobně tak či onak zažehnán, ale určitě nebude poslední. Budou-li vývojáři jádra i nadále slibovat věčně stabilní uživatelské ABI, budou muset řešit všechny jeho aspekty – ne jen systémová volání. Nebude to snadné: moderní systémy vyžadují komplexní komunikační možnosti mezi jaderným a uživatelským prostorem. Ale vývojáři už vyřešili spoustu nelehkých problémů; s ohledem na zvýšenou pozornost, která je věnována zpětné kompatibilitě ABI, je pravděpodobné, že si poradí i s tímto.

Odkazy

[1] http://lwn.net/Articles/193427/
[2] http://kernel.org/pub/linux/kernel/v2.6/testing/ChangeLog-2.6.18-rc3
[3] http://lwn.net/Articles/193154/
[4] http://lwn.net/Articles/193245/
[5] http://lwn.net/Articles/193229/
[6] http://lwn.net/Articles/98379/
[7] http://lwn.net/Articles/193518/
[8] http://lwn.net/Articles/193519/
[9] http://lwn.net/Articles/193526/
[10] http://lwn.net/Articles/14152/
[11] http://www.abclinuxu.cz/slovnik/lkml
[12] http://lwn.net/Articles/193673/
[13] http://lwn.net/Articles/193674/
[14] http://lwn.net/Articles/193675/
[15] http://www.abclinuxu.cz/clanky/jaderne-noviny/jaderne-noviny-26.-7.-2006
[16] http://lwn.net/Articles/172844/
[17] http://lwn.net/Articles/192410/
[18] http://lwn.net/Articles/193656/
[19] http://lwn.net/Articles/193604/
[20] http://lwn.net/Articles/193607/
[21] http://lwn.net/Articles/193613/
[22] http://lwn.net/Articles/193622/
[23] http://lwn.net/Articles/193623/
[24] http://www.abclinuxu.cz/clanky/jaderne-noviny/jaderne-noviny-19.-7.-2006

⋆⋆⋆

Jaderné noviny – 9. 8. 2006

Robert Krátký

Aktuální verze jádra: 2.6.17.8. Citát týdne: Dave Jones. Pohyb ve vývojářské komunitě. Grand Unified Flow Cache. Připojení Linuxu k hypervisorům. Kód nejistého původu.

Aktuální verze jádra: 2.6.17.8

Aktuální verze jádra je 2.6.17.8, vydaná [1] 6. srpna. Tentokrát obsahuje poměrně velký počet důležitých oprav, i když ani v jednom případě nešlo o ⋄CVE [2] problém. *Aktuální předverze je 2.6.18-rc4*, vydaná [3] 6. srpna. Linus k ní napsal: Info najdete v diffu (a v připojené krátké verzi changelogu): hodně malých oprav v různých oblastech – většinou jde o ovladače. Vstupní vrstva, infiniband, usb, síť, zvuk, vlb. Aktualizace cpufreq a architektur. Několik vylepšení auditových pravidel od Ala & Amy.

V síťovacím kódu je také nový mechanismus pro upozorňování na události a funkce (`netdev_alloc_skb()`) pro alokaci paketových bufferů způsobem, který si rozumí s NUMA. Vizte také dlouhý changelog [4]. Aktuální verze -mm stromu je 2.6.18-rc3-mm2 [5]. Mezi nedávné změny patří návrat subsystému CacheFS [6], plná podpora Compact Flash v kódu libata, velká aktualizace x86-64, několik úprav správy paměti a podpora vektorovaného asynchronního I/O.

Citát týdne: Dave Jones

1. zákon o hackování jádra od Davej: Je-li počet iterací, kterými patch projde, vyšší než počet řádků diffu, pravděpodobně nestojí za tu námahu.

– Dave Jones [7]

Pohyb ve vývojářské komunitě

Když Linus oznámil [8] vydání jádra 2.6.18-rc4, přihodil i něco informací navrch:

Po většinu následujících tří týdnů budu převážně off-line (dovolená a pohřeb). Ačkoliv doufám, že budu moci občas aktualizovat svůj strom, požádal jsem Grega KH, aby spravoval git strom a začleňoval užitečné opravy.

Pak bleskurychle zmizel ze scény, aniž by dal -rc4 na kernel.org – opomenutí, které o pár hodin později napravil Greg. Zatímco vývoj jádra bude pokračovat jako obvykle, je pravděpodobné, že se v následujících týdnech dočkáme méně -rc verzí a téměř jistě nevyjde ani finální 2.6.18. Andrew Morton mezitím využil oznámení o 2.6.18-rc3-mm1 [9] k předání vlastních zpráv:

Pro ty, které to zajímá... nedávno jsem se nechal zaměstnat u Google.

Tuto změnu zjevně učinil především kvůli nalezení pracovního prostředí, které mu více vyhovuje; z hlediska vývoje jádra se neočekávají žádné změny. A konečně, Greg Kroah-Hartman oznámil [10] změnu podpory 2.6.16:

Adrian [Bunk] převezme jadernou větev 2.6.16-stable a bude ji spravovat, jak dlouho bude chtít. Bude se držet -stable pravidel zdokumentovaných v souboru Documentation/stable_kernel_rules.txt, ale postará se o strom 2.6.16 o hodně delší dobu, než je stávající stable tým ochoten mu věnovat (my jsme se přesunuli k verzi 2.6.17).

Adrian oznámil své rozhodnutí spravovat toto jádro dlouhodobě už v počátcích vývojového cyklu 2.6.16. Bude zajímavé sledovat, jak se situace vyvine; příprava důležitých patchů pro 2.6.16 bude čím dál těžší úměrně s tím, jak se bude hlavní jádro vzdalovat. Dlouhodobý úspěch projektu může záviset na tom, jestli distributoři takové jádro využijí – a pomohou tak s jeho údržbou.

Grand Unified Flow Cache

Grand Unified Flow Cache [velká sjednocená průtoková keš] je jednou z položek, které se objevují na programech prezentací síťovacích summitů; síťaři asi vědí, co to znamená, ale jsou poněkud laxní, co se vysvětlování toho nápadu týče. Tento koncept se vynořil v souvislosti s diskuzí o síťových kanálech [11] a bylo učiněno množství náznaků, které mi – neb se nebojím vyvozovat mnohé z mála [Jonathan Corbet] – přiblížily význam toho termínu. Kdyby byla implementována, GUFC by znamenala významné změny pro celý síťový stack. Koncept síťových kanálů vyžaduje, aby jádro dokázalo rychle určit cíl každého paketu a hodit jej do správného kanálu. Ještě lepší by bylo, kdyby tuto klasifikaci provedl chytrý síťový adaptér při přijetí paketu, takže by z této části akce bylo jádro úplně vynecháno. Jedním ze způsobů, jak tuto klasifikaci provést, by bylo vytvořit z každého paketu tuple [sada údajů] a ten pak použít jako vyhledávací klíč v nějaké rychlé datové struktuře. Jakmile je v této struktuře (průtokové keši) tuple paketu nalezen, je o jeho osudu rozhodnuto a může být rychle vyprovozen tam, kam má jít. Tento tuple, jak jej popisoval [12] Rusty Russell, by tvořilo sedm parametrů:

- Zdrojová IP adresa.
- Cílová IP adresa.
- Bit indikující, jestli jde o lokální zdroj.
- Bit indikující, jestli jde o lokální cíl.
- Číslo IP protokolu.
- Zdrojový port.
- Cílový port.

Tato čísla jsou dohromady dostatečná k určení spojení, ke kterému každý paket patří. Rychlé vyhledání příchozího paketu by tedy mělo zjistit rozumný cíl (například síťový kanál) bez dalšího zpracovávání. Funkce jako netfilter však tento pěkný obrázek kazí. Netfilter je v rámci jádra nastaven tak, že je každý paket strčen do příslušného řetězce. Když musí každý paket projít stejnou trasou, je výhoda GUFC pryč. Rusty problém popisuje takto:

Chybou (?) netfilteru je, že je úplně obecný: uvidíš všechny pakety, dělej si s nimi, co chceš. Kdybychom místo toho vyžadovali, aby byla všechna pravidla typu „ukaž mi všechny pakety, které odpovídají tomuto tuple“, měli bychom možnost to zkombinovat do jediného vyhledání společně s routováním atd.

Takže řešením by bylo změnit API netfilteru tak, aby lépe spolupracovalo s GUFC. Pravidla by mohla být psána podle vzoru výše uvedeného tuple (s povolenou hvězdičkovou konvencí) a pouze pakety, které by odpovídaly tuplům, by musely projít (pomalou) trasou přes netfilter. To by umožnilo paketům, které filtrovací kód nezajímají, celý mechanismus obejít – a takové rozhodnutí by mohlo být učiněno na základě jediného vyhledávání.

Často by však bylo možné rozhodnout i filtrování paketu přímo na základě tuplu – jak jednou paket tuplu odpovídá, není nutné jej vyhodnocovat ještě s ohledem na pravidlo. Takže když by například kód pro sledování spojení povolil založení nového spojení a tuple popisující takové spojení by byl přidán do keše, nebylo by už pro toto spojení potřeba žádného filtrování. Kdyby netfilter a průtoková keš fungovaly dohromady, šlo by v mnoha případech eliminovat režii spojenou se zpracováním každého paketu.

Jeden ze způsobů, jak zařídit, aby to fungovalo, by bylo mít sadu zpětných volání, která by byla volána pro každý tuple přidaný do průtokové keše. A modul jako netfilter by mohl tuple prohlédnout s ohledem na aktuální sadu pravidel a dát jádru vědět, jestli potřebuje zkoumat pakety, které tuplu odpovídají. Pak by mohly být pakety nasměrovány k příslušným filtrům bez nutnosti využívat v keši tuplů hvězdičkovou konvenci. Celé to má jedno malé mínus:

Samozřejmě to znamená přepsání všech uživatelských nástrojů a dokumentace a vytvoření zcela nové infrastruktury pro sledování spojení a NAT. Ale jestli je to potřeba, tak ať.

Rustyho nikdy dříve podobné překážky nezastavily, takže na to všechno možná opravdu dojde. Ale asi ne moc brzy. Zůstává hodně otázek, na které je potřeba odpovědět před vážným pokusem o implementaci. Jednou z nich je to, jestli by výkon byl skutečně takový, jak lidi doufají; takové plány mohou o dost

zpomalit, jakmile se započítají všechny detaily, které provázejí skutečné nasazení. Aktualizace pravidel by mohla být obtížná; administrátor, který právě změnil pravidla filtrování paketů, nebude chtít trpělivě čekat, až se všechna nová pravidla propracují do keše. Není tak docela jasné, jak přinutit hardware, aby pomáhal s procesem klasifikace. A tak dále. Ale zdá se, že je tu dost zajímavých nápadů, což dříve či později povede k dobrým výsledkům.

Připojení Linuxu k hypervisorům

Paravirtualizace [13] spočívá ve spuštění hostovaného operačního systému v rámci hostitelského systému, přičemž host byl portován na virtuální architekturu, která je *téměř* jako hardware, na kterém ve skutečnosti běží. Tato technika umožňuje spouštění plných hostovaných systémů poměrně efektivním způsobem. Nejznámější svobodnou implementací paravirtualizace zůstává Xen [14]; mezi proprietárními to je VMware. Oba z těchto projektů by rády dostaly alespoň části svého kódu do jádra. Vývojáři však nechtějí začleňovat velkou hromadů háčků [hooks] specifických pro jediné řešení.

Jedním z pokusů o vyřešení tohoto problému je rozhraní VMI [15], které navrhuje VMware. VMI funguje na základě izolace všech operací, které by mohly vyžadovat zásah hypervisora, do speciální sady funkčních volání. Implementace těchto funkcí není zabudovaná v jádře; místo toho jádro při bootu natáhne "hypervisor ROM", která potřebné funkce poskytne. Binární rozhraní mezi jádrem a tímto natahovatelným segmentem je tesáno do kamene, což znamená, že jádra sestavená pro dnešní implementace budou fungovat stejně dobře i se zítřejšími. Tento design zároveň umožňuje, aby jediný binární obraz jádra fungoval s různými hypervisory, nebo – se správnou ROM – v nativním režimu přímo na holém hardwaru. Pevné ABI a možnost natahovat do jádra „binární kusance" [blobs] se však mnoha vývojářům jádra nelíbí. Vypadá to jako další způsob, jak do jádra dostat proprietární kód, což je věc, kterou vývojáři nemají příliš v lásce. Kromě toho, jak říká [16] Rusty Russell:

Správa ABI není naše silná stránka. A bylo by to vyloženě mizerné, kdybychom měli spravovat ABI, i když 99 % z nás by používalo nativní režim, takže bychom si chyb ani nevšimli.

Z tohoto a dalších důvodů nebyla dosavadní cesta VMI do jádra moc uhlazená. To však neodradilo Zachary Amsdena z VMware od snahy o protlačení rozhraní pro binární kód [17]. Proslýchalo se cosi o vývoji alternativního rozhraní pro hypervisory (nazývaném „paravirt_ops"). Raná implementace paravirt_ops [18] byla do ◇LKML [19] poslána 7. srpna, což obrysy trochu objasnilo. Ukázalo se, že paravirt_ops je další struktura plná ukazatelů na funkce – stejně jako mnohé jiné operační struktury používané v jádře. V tomto případě jsou operacemi různé funkce, které většinou vyžadují reakci hypervisora. Jsou mezi nimi věci jako vypínání přerušení, změna kontrolních registrů procesu, změna mapování paměti atd. Jako příklad poslouží jedna z funkcí v paravirt_ops:

```
void (fastcall *irq_disable)(void);
```

Patch také definuje malou funkci, kterou má využívat jádro:

```
static inline void raw_local_irq_disable(void)
{
paravirt_ops.irq_disable();
}
```

Dokud jádro používá pro vypínání přerušení pouze tuto funkci, použije se implementace, kterou poskytl hypervisor. Patch obsahuje sadu operací pro nativní (nevirtualizované) systémy, což způsobí, že se jádro bude chovat jako dříve – či spíše to tak bude, až budou opraveny zbývající chyby. Takové jádro však může být o něco pomalejší, protože mnoho operací, které byly prováděny in-line kódem assembleru, jsou teď zajištěny nepřímým voláním funkce. Pro zmírnění nejhorších dopadů na výkon obsahuje patch paravirt_ops samopatchovací mechanismus k opravení některých volání funkcí – především těch, které se týkají přerušení.

Rozhraní vypadá dost jako VMI; obě umožňují náhradu důležitých nízkoúrovňových operací verzemi z hypervisoru. Rozdíl je v tom, že paravirt_ops je rozhraní založené na zdrojových kódech – nedává žádné záruky binárního rozhraní. Předpokládá se, že toto rozhraní se bude postupem času měnit stejně jako většina interních jaderných rozhraní. A vzhledem k tomu, že jde o relativně novou oblast, je možné, že

paravirt_ops bude jistou dobu ještě více vrtkavé než bývá zvykem. V současné době také neexistuje možnost natahování operací za běhu, takže budou jádra muset být kompilována, aby fungovala s konkrétním hypervisorem.

Na první pohled tedy vypadá paravirt_ops jako konkurent VMI – je na vybranou mezi otevřeným, změnitelným jaderným rozhraním a binárním kódem s pevným ABI. Skutečnost se však má tak, že na paravirt_ops pracuje různorodá skupina vývojářů, včetně zástupců z Xen a, ano, VMware. Části kódu VMI si našly cestu do prvního vydání paravirt_ops. Všichni velcí hráči jsou za tímto vývojem – fakt, který velmi usnadní cestu do jádra. Tak proč pořád chtějí vývojáři VMware binární rozhraní? Vypadá to, že zvažují vytvoření vrstvy spojující paravirt_ops s binárním rozhraním VMI. Tak by byli zcela zodpovědni za správu vlastního ABI, zatímco vývojáři jádra by se mohli po libosti vrtat v paravirt_ops. Někteří ze účastněných vývojářů by byli klidnější, kdyby bylo rozhraní VMI propojeno tímto způsobem – přesto zůstává určitá nedůvěra vůči možnosti linkování ne-GPL binárních modulů.

Vývojáři paravirt_ops by kód rádi dostali do jádra 2.6.19. To se zdá dost smělé vzhledem k tomu, že čas pro začleňování přijde za několik týdnů a v tuto dobu paravirt_ops ještě ani nepobylo v -mm. Jedná se však o funkci, která zcela zmizí, není-li zvolena při konfiguraci, takže to začlenění není tak docela vyloučeno.

Kód nejistého původu

Nedávno byla poslána sada patchů [20] k začlenění do hlavního jádra. Tyto patche využívají (nedokumentovaný) „SMAPI“ BIOS z notebooků Thinkpad pro podporu několika užitečných funkcí Thinkpadů. Vypadá to jako kód, který bývá vítán; zlepšování podpory hardwaru se obecně považuje za dobrou věc. Je tu však jeden problém. Kód byl podepsán takto:

```
Signed-off-by: Shem Multinymous <multinymous@gmail.com>
```

Několik vývojářů rychle poukázalo na to, že taková informace je k ničemu, a že kódu podepsanému zjevným pseudonymem je těžko věřit dost na to, aby mohl do jádra. „Pan Multinymous“ chtěl začlenění dosáhnout následujícími prohlášeními [21]:

Tímto stvrzuji, že patch byl vyvinut výhradně na základě veřejných specifikací, sledování chování hardwaru pomocí metody pokus – omyl a specifikací, které mi byly zpřístupněny v čistém prostoru bez jakýchkoliv podmínek s tím spojených. Takže tento patch je stejně ryzí jako původní ovladač hdaps, který opravuje (a patrně ryzejší než mnoho jiných ovladačů), a ani jediný řádek není odvozenou prací z kódu $JINÝ_OS.

Autor kódu však odmítá prozradit svou identitu, takže ostatní vývojáři odmítají uvažovat o začlenění kódu. Pat by mohl rozlousknout Pavel Machek, který se nabídl, že kód podepíše. Jestli to bude stačit, to pravděpodobně rozhodne Linus, až se vrátí z cest.

Když se ví o SCO, není potřeba být ani moc paranoidní, aby si člověk představil, že by někdo mohl chtít jádro sabotovat pokusem o začlenění ilegálního kódu. Kdyby byl skutečný původ kódu odhalen po jeho rozšíření, mohlo by to z toho být spousta potíží pro vývojáře, distributory a možná i uživatele. Takže je dobře, že vývojáři odolávají pokušení začlenit kód od anonymních přispěvatelů. Epizoda se SCO světu ukázala, jak čistý kód jádra je; rádi bychom, aby takový zůstal.

To je všechno pěkné, ale těžko se vyhnout nepříjemnému pocitu, že kdyby byl tento kód poslán někým s normálně znějícím jménem, nebyl by tolik zkoumán. Kód přeci přichází ze všech koutů světa a nikdo nemá zdroje ani chuť k ověřování, zda ta jména patří skutečným lidem, kteří mají právo daným kódem přispět. Z toho důvodu jsou lidé, kteří posílají kód dokazující velké znalosti nedokumentovaného hardwaru, často tázáni, kde ty znalosti získali. Ověřování odpovědí však může být složité. Obranyschopnost je slabá, ale těžko si představit, jak ji posílit, aniž by byl proces úplně zadušen.

Odkazy

[1] http://lwn.net/Articles/194345/
[2] http://www.abclinuxu.cz/slovnik/cve
[3] http://lwn.net/Articles/194310/
[4] http://kernel.org/pub/linux/kernel/v2.6/testing/ChangeLog-2.6.18-rc4
[5] http://lwn.net/Articles/194316/

[6] http://lwn.net/Articles/100321/
[7] http://kernelslacker.livejournal.com/45863.html
[8] http://lwn.net/Articles/194310/
[9] http://lwn.net/Articles/194314/
[10] http://lwn.net/Articles/194555/
[11] http://www.abclinuxu.cz/clanky/jaderne-noviny/jaderne-noviny-26.-7.-2006
[12] http://lwn.net/Articles/194461/
[13] http://www.abclinuxu.cz/clanky/jaderne-noviny/jaderne-noviny-12.-4.-2006
[14] http://www.abclinuxu.cz/clanky/jaderne-noviny/jaderne-noviny-10.-5.-2006
[15] http://www.abclinuxu.cz/clanky/jaderne-noviny/jaderne-noviny-15.-3.-2006
[16] http://lwn.net/Articles/194551/
[17] http://lwn.net/Articles/194016/
[18] http://lwn.net/Articles/194339/
[19] http://www.abclinuxu.cz/slovnik/lkml
[20] http://lwn.net/Articles/194368/
[21] http://lwn.net/Articles/194731/

⋆ ⋆ ⋆

Jaderné noviny – 16. 8. 2006

Robert Krátký

Aktuální verze jádra: 2.6.18-rc4. Citát týdne: Andrew Morton. Návrat ochrany před zatuhnutím blokového zařízení na síti. Kód (stále) nejistého původu. Rozhraní cdev.

Aktuální verze jádra: 2.6.18-rc4

Aktuální předverze je i nadále 2.6.18-rc4; Linus bude ještě nějakou dobu na dovolené. Mezitím vydal Greg Kroah-Hartman 2.6.18-rc4-gkh1 [1] – obsahuje 64 patchů určených k začlenění do hlavního jádra po Linusově návratu. Aktuální verze -mm stromu je 2.6.18-rc4-mm1 [2]. Mezi nedávné změny patří přepracování konfiguračních voleb SATA (Pokud spustíte naslepo 'make oldconfig', přijdete o disky), nová sada USB funkcí, velká aktualizace x86-64, přepracování kódu protokolu síťového času, podpora –bind připojení pouze pro čtení a nový ovladač embedded řadiče notebooků Thinkpad (přes pochyby o původu – viz níže). *Aktuální 2.4 jádro je 2.4.33*, vydané [3] 11. srpna. To byla poslední verze od Marcela; správu řady 2.4 převzal Willy Tarreau [4].

Citát týdne: Andrew Morton

Ext3 tu bude ještě mnoho let. Nemůžeme ho nechat jen tak shnít kvůli jakési falešné víře, že ho provádění rutinní údržby z nějakého podivného důvodu poškodí.

– Andrew Morton [5]

Návrat ochrany před zatuhnutím blokového zařízení na síti

Právě před rokem se mluvilo o sadě patchů [6] určených k ochraně před zatuhnutím síťového subsystému. Dotyčný problém může nastat ve chvíli, kdy systém používá blokové (diskové) zařízení umístěné na druhé straně síťového spojení. Má-li systém málo paměti, je jednou z věcí, které musí provést, zápis nečistých stránek na disk, což paměť uvolní k využití pro jiné účely. Ale zápis na síťový disk může sám o sobě vyžadovat alokace paměti – potřeba, která se projeví v nejnevhodnější moment. Tento konkrétní problém, který se objevuje i u lokálně připojených disků, je už nějakou dobu vyřešen udržováním kousku paměti rezervovaného speciálně pro blokové I/O operace.

Disky připojené přes síť však mají další problém – žádný zápis nemůže být považován za dokončený, dokud není ze vzdáleného zařízení přijato potvrzení. Přijetí takového potvrzení vyžaduje, aby byl systém schopen přijímat (a zpracovávat) síťové pakety – a to si může vyžádat neomezené množství paměti. Může se objevit spousta příchozích síťových dat, která nebudou mít nic společného s čekajícími I/O požadavky, a tato data mohou znemožnit přijetí paketů, které by systém lapající po paměti tolik potřeboval. Patch pro předcházení zatuhnutí provádí určité změny zaměřené na to, aby systém mohl příchozí bloková I/O data přijímat a zpracovávat vždy. O rok později se ta sada patchů opět vynořila [7]. Původní autor (Daniel Phillips) přenechal hlavní slovo Peteru Zijlstrovi. Nová verze patche v mnoha směrech připomíná své předchůdce, ale přesto je tam dost změn na to, aby stály za bližší pohled.

Hlavní funkci plní patch i nadále pomocí zvětšení pohotovostní rezervované oblasti, kterou spravuje hlavní alokátor stránek. K dispozici je GFP parametr (`__GFP_MEMALLOC`), který v případě potřeby umožňuje uspokojení konkrétního alokačního volání z rezervy. Jde především o to, aby tato rezerva mohla být využívána k přijímání důležitých paketů, aniž by ji zahltily zbytečnosti.

Kvůli tomu také kód, který provádí blokové I/O přes síťové spojení, nastavuje na svých soketech parametr `SOCK_MEMALLOC`. Předchozí verze nastavily parametr na všech připojených síťových rozhraních, aby daly najevo, že daným rozhraním prochází blokové I/O. Současná verze tento krok přeskakuje. Místo toho

si každý pokus o alokaci (paketové) struktury `sk_buff` z ovladače síťového zařízení vezme z paměťové rezervy, je-li to nutné. Takže vydrží-li to rezerva, může systém pro příchozí pakety alokovat buffery.

Klíčem je přijmout důležité pakety, aniž by se rezerva vyčerpala nepotřebnými daty. Proto je síťovací kód opatchován, aby kontroloval parametr `SOCK_MEMALLOC`, jakmile je identifikován soket pro každý příchozí paket. Není-li parametr nastaven a paket využívá paměť z rezervy, bude okamžitě zahozen, aby paměť uvolnil pro jiné využití. Takže pakety týkající se blokového I/O jsou přijaty a zpracovány jako obyčejně, zatímco téměř všechno ostatní je při první příležitosti likvidováno.

Nejčerstvější verze patche obsahuje nový alokátor paměti nazvaný SROG, který je využit pro správu rezervní paměti. Má být rychlý a jednoduchý, aby mohl pro systém co nejrychleji paměť uvolňovat. Kvůli tomu se snaží seskupovat související alokace a pak každou skupinu (většinou strukturu `sk_buff` a s ní spojenou datovou oblast) izolovat na vlastní stránky. Kdykoliv je uvolněn paket, paměť s ním spojená je celá systému hned dostupná.

Vypadá to však, že bude těžké prosadit začlenění patche. Zatuhávání se nezdá příliš pravděpodobné – nekonala se žádná záplava chybových hlášení ohledně této otázky. A ve většině případů lze potížím předejít swapováním na lokální disk. Počet systémů, jejichž majitelé si mohou dovolit fajnová síťová úložná pole, ale zároveň nemohou investovat do lokálního disku pro swapování, není pravděpodobně velký. Rozšíření síťovací vrstvy kvůli řešení tohoto problému se mnoha lidem nelíbí. Správce síťování David Miller by rád viděl [8] jiný přístup k alokacím paměti:

Myslím, že bychom více vytěžili z řešení, které se „síťovou pamětí“ skutečně něco udělá, místo aby říkalo „tahle zařízení jsou odlišná“ nebo „tyhle sokety jsou odlišné“. Zvláštní případy většinou stojí za houby. Už teď v systému globálně limitujeme a kontrolujeme paměť TCP soketů. Kdybychom to dělali pro všechny alokace, což je vlastně výchozí způsob v návrhu alokátoru od Evgenije, mohli bychom ten problém řešit rozumněji.

Komentář odkazuje na patch s alokátorem síťové paměti [9] od Evgenije Poljakova. Jeho práce prochází prudkými změnami, takže je obtížné kód číst. Jádrem je však toto: jde o (další) samostatný alokátor paměti zaměřený na potřeby síťovacího systému. Je navržen tak, aby alokace zůstávaly na jednom lokálním procesoru, takže má každý procesor svou sadu stránek na rozdávání. Alokované objekty jsou uloženy co nejhustěji, aby se minimalizovala interní fragmentace. V současném návrhu není možnost se obrátit na systémový alokátor paměti, takže když některý z procesorů vyčerpá zásobu, alokace selže. Vyčerpání paměti ve zbytku systému však alokátor neovlivní. Autor tvrdí, že to vede ke zvýšení výkonu:

Výkonnostní testy na běžném web serveru založeném na epoll ukázaly znatelné (více než 40 %) zlepšení v počtu požadavků (1600 – 1800 požadavků za vteřinu vs. více než 2300). Lze to přičíst uvolňovacímu algoritmu, který méně namáhá keš, těsnějšímu řazením objektů a tím pádem menší době odezvy z keše, menšímu počtu hledání v keších vyšších vrstev atd.

Kód pamatuje i na mapování síťových bufferů přímo do uživatelského prostoru, které by mohlo fungovat například s budoucí implementací síťových kanálů [10]. Patch (síťový alokátor) rozhodně správce síťování zaujal. Je však dost daleko od případného začlenění a ne všichni jsou přesvědčeni, že vyřeší všechny problémy. Takže jde o diskuzi, která může ještě nějakou dobu pokračovat.

Kód (stále) nejistého původu

Minulý týden [11] jsme se dívali na situaci kolem nového ovladače embedded řadiče u notebooků Thinkpad a jeho autora, který se představil jako „Shem Multinymous“. Vývoj událostí se zastavil, když Pavel Machek navrhl, že pod kód připojí své jméno, což diskuzi trochu utišilo. Ne však na dlouho. Robert Love, autor ovladače akcelerometru, který je (mimo jiné) tímto kódem nahrazován, kód zkontroloval [12]: Jsem rád, že má někdo zjevně lepší přístup k hardwarovým specifikacím než jsem měl já.

Na to reagoval Andrew Morton [13]:

Situace je stále ošemetná. Kde se vzaly ty informace o dalších registrech? [...] Tímto stanovíme precedens a k tomu potřebujeme Linuse. Možná bychom mohli požádat „Shema“, aby svou pravou totožnost soukromě prozradil Linusovi (a možná mně), a na tom dál stavět. Pravidlo by mohlo znít: „každý, kdo se podepíše po kód (Signed-off-by:), by měl znát totožnost ostatních“.

To však nestačí [14] Gregu Kroah-Hartmanovi:

Znamená-li to něco, tak já se těchto patchů ani nedotknu (obyčejně jdou hwmon patche Linusovi přes Jeana a mě). Pokud nechce původní vývojář pracovat otevřeně tak jako my, respektuji to, ale nemohu přijmout riziko vyplývající z akceptování takového kódu.

Jean Delvare také odmítl kód kontrolovat [15], protože právní pochybnosti jsou příliš velké. Shem Multinymous však vypadá, že by byl ochoten prozradit své jméno Linusovi a Andrewovi, pokud by to dostalo kód do jádra. Takže je možné, že by to tak mohlo dopadnout – kód by přeskočil správce, kteří by jej normálně měli na starosti. Všechna nejistota by však asi nezmizela, což by mohlo vést i ktomu, že by distributoři vynechali daný kód z jader, která dodávají.

"Shem" také poslal dvě [16]samostatné [17] zprávy ohledně původu informací použitých v ovladači. Jak se zdá, tak příběh začíná reverse-engineeringem ovladače pro Windows. Pak byla objevena pravá specifikace čipu embedded řadiče. A pak už šlo hlavně o poskládání kousků dohromady. Tak je to alespoň podáváno.

Pokud příběh obstojí, mohl by být pravděpodobně nový kód začleněn do jádra bez obav; měl by být přinejmenším tak legální jako ovladač, který by nahradil. Ale pokud se do jádra dostane, stanoví tak určitý precedens: anonymní příspěvky (nebo aspoň ty, které jsou podepsány zjevným pseudonymem) to budou mít těžké. Nikdo nechce být tím, kdo do jádra uvedl nežádoucí kód.

Rozhraní cdev

Od nepaměti bylo základní registrační rozhraní pro znaková zařízení v jádře následující:

```
int register_chrdev(unsigned int major, const char *name,
                    const struct file_operations *fops);
int unregister_chrdev(unsigned int major, const char *name);
```

V dávných dobách alokovalo `register_chrdev()` všech 256 minor čísel spojených s daným `major` a zadané `name` a `file_operations` byly přiřazeny ke každému z nich. Je-li jako major číslo zadána nula, bude alokováno za běhu. Odpovídající volání `unregister_chrdev()` všechna minor čísla uvolní. Toto volání požadovalo `name` z bezpečnostních důvodů; pokud neodpovídalo hodnotě udané při registraci major čísla, volání selhalo.

Během rušného období před vydáním jádra 2.6.0 se Al Viro rozhodl přijít na způsob, jak zvětšit rozsah čísel pro zařízení. Jedním z problémů vyžadujících řešení bylo obrovské množství ovladačů, které "věděly", že minor čísla nikdy nejsou vyšší než 255. Jednou možností bylo prověřit každý ovladač v jádře, jestli se k minor číslům chová správně. Času však nebylo nazbyt a dobrovolníků pro takovou práci ještě méně. Takže na to šel Al jinak: vytvořil pro registraci znakových zařízení nové rozhraní a pak reimplementoval staré rozhraní jako vrstvu zajišťující kompatibilitu, která bude alokovat pro dané major číslo minor čísla od 0 do 255. Tak mohl neupravený kód fungovat jako dosud, protože jádro zaručovalo, že nikdy nespatří minor čísla, která neviděl dříve. Postupem času měly být ovladače převedeny na nové rozhraní, které má množství výhod.

Dopadlo to však tak, že se převedení nikdy nekonalo. Protože staré rozhraní i nadále fungovalo, lidi ho znali a bylo trochu jednodušší jej používat, vývojáři u něj zůstali. Ještě podstatnější možná bylo, že nikdy nedošlo na obávaný nedostatek čísel zařízení. Větší využití dynamických čísel, obecnější rozhraní pro zařízení a mechanismus hotplug dohromady zajistily, že se (většina) linuxových systémů do staršího číselného prostoru snadno vešla, takže rozšířená čísla byla využívána zřídka. Pohled na můj systém odhalí přesně tři minor čísla větší než 255, všechna v `/dev/bus/usb`. Takže nikdy nebyl důvod přecházet na nové rozhraní.

Alexey Dobriyan si nedávno všiml, že `unregister_chrdev()` už nekontroluje parametr `name`, takže poslal patch [18], který ten argument odstranil a při té příležitosti opravil všechny „volající“ (vše, co funkci volalo). Jonathan Corbet poznamenal, že by to mohla být ta správná chvíle pro převod na nové rozhraní – místo přepracovávání staršího rozhraní pro kompatibilitu. Jiný vývojář reagoval s tím, že by bylo fajn mít pro nové rozhraní lepší dokumentaci. Takže tady je rychlý přehled toho, jak by měla v 2.6 vypadat registrace znakových zařízení.

Novější rozhraní rozděluje registraci znakových zařízení na dva kroky: alokace rozsahu čísel zařízení a přiřazení specifických zařízení k těmto číslům. Alokační fázi zajišťuje jedna z následujících funkcí:

```
int register_chrdev_region(dev_t first, unsigned int count,
                           const char *name);
int alloc_chrdev_region(dev_t *first, unsigned int firstminor,
                        unsigned int count, char *name);
```

První způsob alokuje `count` minor čísel začínajících u páru major/minor určeném ve `first` a zapamatuje u každého `name`. Druhý způsob je určen pro použití v případech, kdy major číslo neznáme dopředu; alokuje major číslo, pak alokuje `count` minor čísel začínajících na `firstminor`. Začátek alokovaného rozsahu čísel bude vrácen ve `first`. Návratová hodnota bude nula při úspěchu a záporný chybový kód při selhání.

Několik věcí stojí za zmínku. U obou možností může být použité major číslo sdíleno s jinými, úplně nesouvisejícími zařízeními. Pouze specifikovaný rozsah alokovaných minor čísel patří daným volajícím. Tato minor čísla mohou být větší než 255. Je možné, že alokovaný rozsah čísel zařízení přeteče rozsah minor čísel do dalšího major čísla. To je záměr a vše by mělo správně fungovat – ačkoliv nevím o žádném produkčním jádře, kde by alokace fungovaly tímto způsobem. Bez ohledu na to, která alokační funkce byla použita, mohou být čísla zařízení systému vracena takto:

```
void unregister_chrdev_region(dev_t first, unsigned int count);
```

Přiřazení čísel zařízení ke konkrétním zařízením se děje prostřednictvím struktury `cdev`, která se nachází v `<linux/cdev.h>`. Strukturu `cdev` je možné inicializovat následující sekvencí:

```
struct cdev *my_dev = cdev_alloc();

if (my_dev != NULL)
my_dev->ops = &my_fops;  /* The file_operations structure */
my_dev->owner = THIS_MODULE;
else
/* No memory, we lose */
```

Při běžném použití však bude struktura `cdev` začleněna v nějaké větší struktuře specifické pro dané zařízení a alokace proběhne v rámci této struktury. V takovém případě bude funkce pro inicializaci `cdev` vypadat takto:

```
void cdev_init(struct cdev *cdev, const struct file_operations *fops);
/* Need to set ->owner separately */
```

Oběma způsoby je struktura uvedena do řádného provozního stavu a bude vybavena `file_operations`, které budou volány z přiřazeného zařízení. Pole `owner` by mělo být inicializováno jako `THIS_MODULE`, aby se zabránilo neuváženému odstranění modulu v případě, že by zařízení bylo ještě aktivní.

Posledním krokem je přidání `cdev` do systému a přiřazení k příslušným číslům zařízení. Nástrojem pro tuto práci je:

```
int cdev_add(struct cdev *cdev, dev_t first, unsigned int count);
```

Funkce přidá `cdev` do systému. Bude se starat o operace pro `count` čísel zařízení začínajících na `first`; `cdev` bude často sloužit jedinému číslu zařízení, ale není to podmínkou. `cdev_add()` může selhat; je-li návratový kód nula, zařízení *nebylo* do systému přidáno.

A stejně důležité je to, že jakmile `cdev_add()` uspěje, zařízení je zpřístupněno a jeho operace se soubory mohou být volány jádrem. Takže ovladač by `cdev_add()` neměl volat, dokud není inicializace přiřazeného zařízení dokončena. Jinak dojde k nepříjemnostem. Odstranění znakového zařízení ze systému se provede pomocí:

```
void cdev_del(struct cdev *cdev);
```

Po tomto volání by se na `cdev` nemělo odkazovat. Zvláště pokud byla `cdev` získána pomocí `cdev_alloc()`, je pravděpodobné, že bude uvolněna v `cdev_del()`. Jeden poslední trik, který se vyplatí znát: když jsou volány souborové operace na znakovém zařízení, bude přiřazený `inode` ukazatel předán jako obvykle. Pole `inode->i_cdev` obsahuje ukazatel na strukturu `cdev` pro dané zařízení. Ovladače ten ukazatel mohou používat k získání vlastní struktury specifické pro zařízení (např. s `container_of()`). Není už tedy nutné se pokoušet mapovat minor číslo na interní zařízení – operace, která se mnoha ovladačům nedařila. Rozhraní `cdev` se během raného vývoje 2.6 trochu změnilo, ale teď už se s ním nějakou dobu nic neděje.

Odkazy

[1] http://lwn.net/Articles/195094/
[2] http://lwn.net/Articles/195269/
[3] http://lwn.net/Articles/195017/
[4] http://www.abclinuxu.cz/clanky/jaderne-noviny/jaderne-noviny-2.-8.-2006
[5] http://lwn.net/Articles/195801/
[6] http://lwn.net/Articles/146689/
[7] http://lwn.net/Articles/195308/
[8] http://lwn.net/Articles/195430/
[9] http://lwn.net/Articles/195292/
[10] /clanky/jaderne-noviny/jaderne-noviny-26.-7.-2006
[11] /clanky/jaderne-noviny/jaderne-noviny-9.-8.-2006
[12] http://lwn.net/Articles/195648/
[13] http://lwn.net/Articles/195649/
[14] http://lwn.net/Articles/195652/
[15] http://lwn.net/Articles/195653/
[16] http://lwn.net/Articles/195654/
[17] http://lwn.net/Articles/195655/
[18] http://lwn.net/Articles/195617/

⋆⋆⋆

Zprávičky

Obliba OpenDocument Format stoupá

Obliba OpenDocument Format (ODF) stoupá, k podpoře tohoto formátu se připojilo již 280 organizací. Po Belgii a Francii padlo i v Malajsii rozhodnutí, že ODF bude preferovaným formátem ve veřejném sektoru.

→Luboš Doležel ⋆ *1.8.2006*

SenoCMS 1.0 – OpenSource publikační systém

SenoCMS (Search ENgine Optimized Content Management System) je k dispozici jako funkční studie OpenSource publikačního systému zaměřeného na SEO a konformitu se standardy W3C. Vznikl jako projekt k diplomové práci studentky rakouské Fachhochschule Hagenberg a podle všeho čeká na komunitu dalších vývojářů a uživatelů (kontakt na tvůrkyni Stefanie Poltschak). Dokumentace je zatím k dispozici pouze v němčině.

→Martin Tesař ⋆ *1.8.2006*

Omezování vzdáleného přihlášení u OpenSSH

Povolit všem uživatelům v systému vzdálený přístup přes SSH nemusí být z bezpečnostních důvodů dobrý nápad. Na nixCraftu se dočteme, jak zakazovat či povolovat vzdálené přihlášení skupinám nebo konkrétním uživatelům.

→Luboš Doležel ⋆ *1.8.2006*

Poznáváme User Mode Linux

InformIT představuje User Mode Linux (UML) jako port Linuxu pro Linux. Srovnává jeho vlastnosti s VMware, BSD Jail, nebo dokonce s chrootem. Vysvětluje, proč zvolit právě UML, k čemu se běžně používá, ale také se dočtete o jeho historii.

→Luboš Doležel ⋆ *1.8.2006*

Oprava Kopete 0.12.1 pro ICQ protokol

Jistě každý, kdo používá Kopete (v. 0.12.1) pro komunikaci přes ICQ, zjistil, že mu klient opět hlásí chybovou hlášku při připojování (ICQ server se domnívá, že váš klient je příliš starý). Řešením je aplikovat patch na zdrojové kódy a znovu je přeložit. Zprávičku jsem napsal na základě diskuse na kdeforum.org.

→Michal Wirth ⋆ *1.8.2006*

Balíčkovací systém Smart v SUSE LINUX

Na serveru suseportal.cz vyšel návod, který vás provede instalací a používáním balíčkovacího systému Smart v operačním systému SUSE LINUX. Z oficiálních stránek si pak můžete stáhnout balíček i pro vaši distribuci a ocenit tak výhody tohoto nástroje.

→Jiří Větvička ⋆ *1.8.2006*

200 000 000 stažení Firefoxu

Projekt Spread Firefox oznámil, že počet stažení Firefoxu překročil 200 000 000.

→Jindřich Pozlovský ⋆ *1.8.2006*

Povídání o mýtech okolo linuxového jádra

Greg Kroah-Hartman vyvracel na letošním „Linux Symposium“ některé mýty ohledně linuxového jádra, jako například mýty ohledně podpory některých zařízení a myšlence binárních ovladačů přímo v jádře. Celou řeč najdete na kroah.com. Minimálně kvůli pěkným slidům to stojí za přečtení.

→Jiří „Geo“ Lužnický ⋆ *1.8.2006*

Snadnější správa hudby na iRiver H300

Ian Douglas napsal několik skriptů pro zjednodušení spravování hudby na přehrávačích iRiver H300. Po zběžném prozkoumání se zdá, že by měly být funkční s libovolným přehrávačem podporujícím playlisty. Nejvíce je ocení zejména ti, kteří si s oblibou tvoří vlastní hudební kompilace.

→Jiří „Geo“ Lužnický ⋆ *1.8.2006*

Ukázka z knížky Ubuntu Hacks

Pro získání představy o stavu linuxové literatury pro mírně pokročilé uživatele si lze přečíst na serveru Linux.com úryvek z knížky Ubuntu Hacks o Power Managmentu.

→Jiří „Geo“ Lužnický ⋆ *1.8.2006*

KDE 3.5.4

Ač ještě neohlášeno, v repozitářích SUSE se objevilo KDE 3.5.4. Stejně tak je na ftp.kde.org/pub/kde/stable verze i pro Kubuntu a další distribuce.

→Juraj Václavík ⋆ *2.8.2006*

KDE 3.5.4 oficiálně ohlášeno

Krátce po umístění do repozitářů bylo oficiálně představeno nové KDE 3.5.4. Jedná se o opravnou verzi, celkem bylo odstraněno více než 100 chyb v aplikacích Konqueror, Kate, Konsole, KOrganizer a dalších.

→Martin „mhb“ Böhm ⋆ *2.8.2006*

Mozilla Thunderbird 2 Alpha 1

Mozilla nedávno vydala Mozilla Thunderbird 2 Alpha 1. Na NewsForge se můžete dočíst, co tato verze přináší uživatelům nového.

→Luboš Doležel ⋆ *2.8.2006*

Tony Mobily: s Red Hat to půjde z kopce

Tony Mobily ve svém článku píše, proč si myslí, že to s Red Hat půjde z kopce. Red Hat prý zanevřel na deskopové uživatele a mnoho administrátorů bude brzy stát před rozhodnutím: Red Hat Linux nebo Ubuntu Server?

→Luboš Doležel ⋆ *2.8.2006*

Používáme HAL a Ivman

V článku na Linux.com se můžete dočíst, jak vyvolávat softwarové události na základě hardwarových změn pomocí HAL a Ivman. Dozvíte se, jak nechat automaticky připojit USB disk, nebo třeba jak zhibernovat systém při stisku příslušného tlačítka.

→Luboš Doležel ⋆ *2.8.2006*

GNU Cash 2.0.1

Vyšla nová verze (2.0.1) programu GnuCash, který má za úkol pomoci vám se správou (nejen) domácích financí. Nová verze opravuje mnoho malých, ale otravných chyb. Viz GnuCash – domácí účetnictví na úrovni.

→Jiří „Geo“ Lužnický ⋆ *2.8.2006*

Jak na knihy z projektu Gutenberg

Server Newsforge.com radí, jak na čtení elektronických knih z projektu Gutenberg. Představuje open source programy pro PalmOS a Pocket PC a dále zmiňuje, jak se vypořádat s jistými problémy ve formátování textu při používání těchto programů.

→Jiří „Geo“ Lužnický ⋆ *2.8.2006*

Zpráva z 2006 Linux Symposium

Server Newsforge.com dnes přinesl rozsáhlou zprávu z akce Linux Symposium [zprávička]. Zpráva popisuje, o čem byly nejzajímavější přednášky, a je doplněna o videozáznamy rozhovorů s některými přednášejícími.

→Jiří „Geo“ Lužnický ⋆ *2.8.2006*

Tvorba e-booků s open source nástroji

Rádi čtete knihy, ať už jste kdekoliv, ale ty tradiční vám připadají nepraktické? Pokud máte PDA, mohla by vás zaujmout příprava e-booků pro tato zařízení pomocí open source nástrojů.

→Luboš Doležel ⋆ *3.8.2006*

SeaMonkey 1.0.4

Vyšla nová verze prohlížeče SeaMonkey 1.0.4. Byl vyřešen problém s pluginy z minulé verze a bylo opraveno několik bezpečnostních chyb.

→Martin B. ⋆ *3.8.2006*

Správa a monitoring sítí s OpenNMS

OpenNMS je nástroj pro správu a monitoring velkých podnikových sítí, který umožňuje získávat informace od vzdálených strojů pomocí SNMP. Instalaci a konfiguraci tohoto software vám vysvětlí návod od HowtoForge.

→Luboš Doležel ⋆ *3.8.2006*

Novell přestal distribuovat proprietární moduly

Novell spolu s SUSE Linux Enterprise Server 10 přestal distribuovat proprietární moduly pro jádro, jako jsou např. ovladače grafických karet. Ztotžňuje se tím s názorem Free Software Foundation, že tyto moduly jsou přinejmenším neetické.

→Luboš Doležel ⋆ *3.8.2006*

PHP 4.4.3

Po delší době vyšla další verze PHP 4. Tato verze 4.4.3 především opravuje řadu chyb (Changelog).

→Stanislav Petr ⋆ *3.8.2006*

Přestřelka mezi RH a Ubuntu

Na serveru Freesoftwaremagazine.com vyšel krátký článek, který se opírá do Ubuntu Linuxu. Ten Ubuntu vyčítá, že postupně pohřbívá Red Hat Linuxu pomocí zmenšování jeho uživatelské základny. Na to samozřejmě reagoval Mark Shuttleworth ve svém blogu, kde uvádí některá tvrzení na pravou míru.

→Jiří „Geo“ Lužnický ⋆ *3.8.2006*

Konference Black Hat

Včera začala v Las Vegas konference Black Hat zaměřená na počítačovou bezpečnost. Autor článku na Newsforge.com si nejdříve stěžuje na dlouhé fronty na cokoliv. Poté už jen zmiňuje některá zajímavá témata, která se probírala. My, kdo nemáme to štěstí přímo se zúčastnit, můžeme počkat akorát na video záznamy.

→Jiří „Geo“ Lužnický ⋆ *3.8.2006*

Vyšel Firefox 1.5.0.6

Na stránkách Firefoxu se objevila nová verze 1.5.0.6 tohoto prohlížeče. Ta opravuje (stejně jako nová verze prohlížeče Seamonkey [zprávička]) chybu při přehrávání videa přes protokol MMS.

→Jiří „Geo“ Lužnický ⋆ *3.8.2006*

Xournal – psaní poznámek od ruky

Zatímco psaní obyčejného textu na klávesnici bude rychlejší než psaní textu rukou, nemusí tomu tak být, pokud např. zapisujeme matematické vzorce. V open source světě se řešení pro psaní s tabletem nazývá Xournal, a právě s ním vás seznámí tento článek.

→Luboš Doležel ⋆ *4.8.2006*

Rozhovor o stavu OpenOffice.org

Článek na NewsForge přináší rozhovor s Louis Suárez-Potts, který je předsedou komunitní rady OpenOffice.org. Baví se o posledních vylepšeních v tomto software, rozšířeních jako u Firefoxu nebo problémech s přenositelností maker.

→Luboš Doležel ⋆ *4.8.2006*

TransGaming zrazuje uživatele GNU/Linuxu?

Jeden z hráčů na Linuxu se rozepsal o pravděpodobném budoucím dění ve firmě TransGaming, která stojí za známou Cedegou. Tvrdí, že peníze získané za předplatné Cedegy tato firma využívá na vývoj emulace pro Mac OS X a znovu se tak staví k Linuxu zády. Vyzývá k podpoře nativních portů.

→Luboš Doležel ⋆ *4.8.2006*

Jak na firewall v *BSD

Server OnLamp vydal článek o konfiguraci osobního firewallu pod *BSD pomocí pf. K tvorbě pravidel autor článku používá program fwbuilder. Vše ostatní je řešeno na příkazové řádce (stejně jako ukázka vytvořených pravidel).

→Jiří „Geo“ Lužnický ⋆ *4.8.2006*

BlackHat, den druhý

Konference BlackHat [zprávička] pokračovala druhým dnem. J. Barr v článku na NewsForge přinesl reportáž i z tohoto dne. Na pořadu bylo mimo jiné povídání o situaci okolo spyware (a to jak z technického, tak i právnického hlediska), ženy v IT a rootkity.

→Jiří „Geo“ Lužnický ⋆ *4.8.2006*

RSS vybraných blogů na AbcLinuxu

Po nesčetných žádostech obsahuje AbcLinuxu blogDigest, což je RSS kanál s blogy, které jsou zaměřené na Linux, Unix, nebo na IT. Více v oznámení v Leošově blogu.

→Michal Vyskočil ⋆ *4.8.2006*

Cold War a Gorky 17 k dostání v ČR

Na portálu www.SoftPortal.cz začali prodávat spoustu her od LGP, mezi nimiž nechybí Gorky 17 a skvělý Cold War.

→Jiří Němec ⋆ *4.8.2006*

Vyšel gDVDshrink 0.3-9

Vyšla nová verze programu gDVDshrink (0.3-9), který slouží k „smrsknutí“ videa z DVD, tak aby se vešlo na jedno jednovrstvé DVD médium. Nová verze hlavně přidává podporu titulků, a dále přináší drobné zrychlení.

→Jiří „Geo“ Lužnický ⋆ *4.8.2006*

Co je a jak funguje initrd

Zajímá vás, co je to initrd (initial RAM disk), jaký má význam a jak funguje? Článek od IBM vám vše detailně vysvětlí, a to včetně jeho používání.

→Luboš Doležel ⋆ *7.8.2006*

Základy síťování od Linux Mint

Linux Mint popisuje základy síťování na Linuxu. Ve čtyřech článcích se naučíte používat statické i dynamické IP adresy, filtrování MAC adres a ukládání nastavení do konfiguračních souborů.

→Luboš Doležel ⋆ *7.8.2006*

Lenovo bude nabízet předinstalovaný SUSE Linux

Po dřívějších protichůdných zprávách o (ne)podpoře Linuxu ze strany Lenovo, výrobce notebooků ThinkPad, dala tato společnost definitivně najevo, jaká bude skutečnost. Lenovo bude na svých noteboocích nabízet možnost předinstalace SUSE Linux Enterprise Desktop 10.

→Luboš Doležel ⋆ *7.8.2006*

Fedora Core 6 test 2

Venku je druhá testovací verze distribuce Fedora Core 6. Stahujte z torrent.fedoraproject.org. Novinkou je fungující Java applet pro Firefox přímo v instalaci a nové fonty DejaVu.

→Radek Vokál ⋆ *7.8.2006*

Open Source sbližuje vývojáře s uživateli

Na serveru Freesoftwaremagazine.com vyšel článek od Terryho Hancocka o stále více se rozevírajících nůžkách mezi programátory a uživateli. V open source vidí možnost, jak tuto díru zmenšit, protože umožňuje uživateli účastnit se vývoje.

→Jiří „Geo“ Lužnický ⋆ *7.8.2006*

Google Internals

Zajímáte-li se o Google, pak možná víte, že jejich stroje (údajně jich je 450 000) běží na Red Hat Linuxu se staršími jádry (2.0, 2.2). Pokud se chcete dozvědět něco více, pak si můžete prostudovat slidy od Tobyho DiPasquala ze setkání filadelfské LUG.

→Jiří „Geo“ Lužnický ⋆ *7.8.2006*

Intrusion Detection System OSSEC-HIDS

Zdá se vám, že používat Chkrootkit, je příliš málo na zabezpečení vašeho domácího stroje proti rootkitům a jinému malwaru? Článek na serveru Linux.com představuje OSSEC-HIDS, což je open source řešení pro IDS („detekci narušení systému“).

→Jiří „Geo“ Lužnický ⋆ *7.8.2006*

Tom's Hardware o Linuxu na notebooku

Tom's Hardware přináší rozsáhlý článek o „linuxizaci" notebooků. Chcete-li sebe nebo kamarády přesvědčit o tom, že Linux si bude s notebookem rozumět, začtěte se. Srovnávány jsou distribuce Fedora Core 5 a openSUSE 10.1.

→Robert Krátký ⋆ *8.8.2006*

Srovnání NPTL a LinuxThreads

Na webu developerWorks provozovaném IBM se srovnávají vlastnosti modelů vláken NPTL a LinuxThreads, tedy jejich přednosti a omezení. Dozvíte, jak se liší jejich podpora mezi některými předními distribucemi.

→Luboš Doležel ⋆ *8.8.2006*

Zpracování RAW obrázků na Linuxu

Pokud vlastníte fotoaparát s možností ukládání do formátu RAW, možná vás zaujme článek o podpoře této skupiny formátů na Linuxu. Dočtete se, jaké různé RAW formáty existují a čím je můžete zpracovávat.

→Luboš Doležel ⋆ *8.8.2006*

Tvorba webových bannerů bez použití GUI

Pokud chcete vytvořit webový banner, ale vystačíte si s jednoduchým vzhledem, mohli byste zkusit nějaký udělat i bez použití grafického rozhraní. Článek od Linux.com vám poradí jak na to.

→Luboš Doležel ⋆ *8.8.2006*

Jak na program Remind

Zapomínáte často různá výročí, schůzky a jiné podobné věci. Článek na Linux.com vám poradí, jak se toho vyvarovat pomocí programu Remind. Je popsáno nejen jak program ovládat pomocí různých grafických frontedů, ale seznámí vás i s jeho ovládáním pomocí příkazové řádky (přímou editací souborů).

→Jiří „Geo" Lužnický ⋆ *8.8.2006*

Testování Mozillích kalendářů Sunbird a Lightning

Jelikož se blíží vydání aplikací Sunbird a Lightning (kalendáře), vyzývá Mozilla uživatele k jejich testování. Žádné předchozí zkušenosti s testováním nejsou vyžadovány. Jak na to, si můžete přečíst na stránkách wiki.mozilla.org.

→Jiří „Geo" Lužnický ⋆ *8.8.2006*

OpenZaurus 3.5.4.2-rc0

OpenZaurus je alternativní OS pro Zaurus. Verze 3.5.4.2-rc0 obsahuje jádro řady 2.6. Nově jsou podporovány modely sl-5600 (poodle) a sl-6000 (tosa). Další přibudou brzy v rc1. Více se dozvíte v poznámkách k vydání.

→Jiří „Geo" Lužnický ⋆ *8.8.2006*

Fedora Core 4 přesunuta do Fedora Legacy

Fedora Core 4 byla (po vydání Fedora Core 6 Test 2) přesunuta do správy projektu Fedora Legacy, který bude od této chvíle zodpovědný za vydávání bezpečnostních oprav a opravných aktualizací. Zároveň byl oznámen definitivní konec podpory distribucí Fedora Core 1 a 2.

→Robert Krátký ⋆ *9.8.2006*

Automatizované garáže a OSS?

O tom, aké problémy spôsobili v automatizovaných garážach v New Jersey komerčné licencie a bomby v softvéri – článok na Wired News.

→White Raven ⋆ *9.8.2006*

Dopravní nehoda

Nejen lidé bývají obětmi dopravních nehod. V americkém Texasu při havárii nákladního automobilu patřícího zoologické zahradě přišli o život čtyři tučňáci a pátý má zlomené křídlo, píše idnes.

→Tomáš Hála ⋆ *9.8.2006*

Interview s Timothee Bessetem z ID Software

Na serveru LinuxGames vyzpovídali Timothee „TTimo" Besseta, linuxového experta z ID Software. Baví se o editoru GtkRadiant, rychlosti vydávání linuxových klientů her, zdrojových kódech nebo také ovladačích.

→Luboš Doležel ⋆ *9.8.2006*

Přibližujeme free software těm nejmladším

Pokud chcete, aby si i vaše ratolesti oblíbily nějaký ten svobodný software, určitě nepřehlédněte článek, který se tématem her a jiných programů pro malé zabývá. Zmiňuje se o programech pro prostředí KDE i GNOME.

→Luboš Doležel ⋆ *9.8.2006*

Konflikt mezi Xenem a VMware

VMware, stejně jako Xen, má v plánu poskytovat virtualizaci založenou na principu hypervizoru. Bohužel, oba dva vyžadují spolupráci ze strany kernelu, ale každý to chce řešit po svém. O celém konfliktu si můžete přečíst na serveru InfoWorld.

→Luboš Doležel ⋆ *9.8.2006*

Finální verze Freespire

Dne 7. srpna 2006 vyšla finální verze distribuce Freespire 1.0, která slibuje legální podporu pro MP3,

DVD, Windows Media a mnoho dalších formátů. Stahovat je možné přes BitTorrent nebo z odkazů na stránkách projektu. Zájemci mohou stahovat i čistě OSS variantu bez proprietarních programů.
→Libor Daněk ⋆ *9.8.2006*

KTorrent 2.0

Vyšla finální verze BitTorrent programu pro KDE KTorrent 2.0. Nová verze přináší mimo jiné podporu DHT a šifrovaného přenosu.
→honya ⋆ *9.8.2006*

Bilboard s reklamou na Ubuntu spatřen v USA

Podle fotky bilboardu, která byla pořízena u jedné z dálnic v Kalifornii, to vypadá, že někdo uspořádal pro Ubuntu reklamní kampaň. Vzhledem k tomu, že je znám autor fotky, tak by se nemuselo jednat o podvrh.
→Jiří „Geo“ Lužnický ⋆ *9.8.2006*

Umit, nový frontend pro Nmap

Program Nmap (nejen) pro skenování portů nejspíše znáte. Možná, že víte i o existenci frontendu Nmapfe. Ten se ale už někomu může zdát ošklivý a starý. Nový frontend Umit (používající Python a GTK) přináší zajímavé možnosti, jako například tvorbu profilů pro skenování a podobně.
→Jiří „Geo“ Lužnický ⋆ *9.8.2006*

Měření výkonu aplikací pomocí NetBeans Profileru

Často se tvrdí, že aplikace napsané v Javě spotřebovávají více systémových prostředků, než by bylo vhodné. Pokud potřebujete zjistit, jak konkrétně na tom je vaše aplikace, můžete zkusit použít NetBeans Profiler. Můžete inspirovat návodem z Netbeans.org.
→Jiří „Geo“ Lužnický ⋆ *9.8.2006*

DRI ovladač pre integrovanú grafiku Intel 965

Intel vydal open source 2D Xorg a DRI 3D driver pre svoju novú integrovanú grafiku v 965 Express chipsete. Viac podrobností na IntelLinuxGraphics.org.
→pzad ⋆ *9.8.2006*

Ubuntu 6.06.1 – opravuje přes 300 chyb

Ubuntu 6.06.1 je link na opravnú verziu Ubuntu s vyše 300 opravami. Ťahajte, ak plánujete inštalovať práve teraz.
→Peter Kotrcka ⋆ *10.8.2006*

Velký návrat Apple k open-source?

Apple překvapil open-source svět: otevřel zdrojové kódy jádra pro procesory od Intelu i PPC, dále otevřel iCal server, Bonjour a Launchd a spustil komunitní portál věnovaný těmto aktivitám. Více v Apple Mailing Lists.
→Daniel Kvasnička ml. ⋆ *10.8.2006*

Je svobodný software komunistický? Možná ano...

Free software je často obviňován z inklinace ke komunismu či socialismu. Terry Hancock se rozhodl postavit se k tomuto nařčení jiným způsobem: připustil, že je na něm něco pravdy. Více na Free Software Magazine.
→Daniel Kvasnička ml. ⋆ *10.8.2006*

Port knocking se službou knockd

Zajímavým způsobem ochrany systému proti útočníkům je port knocking. Díky službě knockd můžete tuto techniku na svém serveru používat i bez složitých skriptů, stačí jen nastavit parametry.
→Luboš Doležel ⋆ *10.8.2006*

Proč se GPL v3 zabývá DRM

Světem open source stále hýbou debaty o licenci GPL v3, obzvláště po nedávném vydání druhého návrhu. Pravděpodobně nejspornější částí licence je boj proti DRM. Pokud vás zajímá, proč by GPL mělo problém DRM řešit, přečtěte si některé důvody.
→Luboš Doležel ⋆ *10.8.2006*

Komunitní SUSE Linux už jen jako openSUSE

Novell se rozhodl přejmenovat SUSE Linux. Ten už bude označován pouze názvem openSUSE a bude tak odrážet jméno projektu openSUSE.org. Enterprise varianta SUSE Linux Enterprise zůstane pod původním názvem.
→Luboš Doležel ⋆ *10.8.2006*

LiveCD Gparted 0.2.5-4

Vyšlo LiveCD Gparted 0.2.5-4. Přináší několik změn k pohodlnějšímu používání. První z nich je automatická detekce grafické karty. Dále přibyla možnost nahrát celé CD do paměti počítače, a uvolnit tím CD mechaniku. Celý seznam změn. Stahovat můžete ze sourceforge.net (ISO).
→Jiří „Geo“ Lužnický ⋆ *10.8.2006*

FBReader 0.7.4h – opraveny nepříjemné chyby

Vlastníte-li PDA s prostředím OPIE, pak vás jistě potěší nová verze programu FBReader. Ve verzi 0.7.4h byly vyřešeny problémy s rotací displeje a nástrojovou lištou. Varianta pro stolní počítače stále zůstává v experimentálním stádiu.
→Jiří „Geo“ Lužnický ⋆ *10.8.2006*

Wine 0.9.19

Vyšlo Wine 0.9.19. Přináší vylepšení IDL kompilátoru, lepší podporu pro FreeBSD a opravuje spoustu chyb.

→David Watzke ⋆ *10.8.2006*

Recenze distribuce Freespire 1.0

Server Linuxformat se podíval na zoubek distribuci Freespire 1.0 [zprávička]. Zamýšlí se například nad tím, jestli právě podpora médií tak, jak ji předvádí Freespire, může této distribuci zaručit vysoké umístění mezi ostatními distribucemi.

→Jiří „Geo“ Lužnický ⋆ *10.8.2006*

Coverity pomáhá hledat chyby ve Firefoxu

Coverity, komerční software pro hledání chyb ve zdrojovém kódu, je po Linuxu, MySQL a X.org bezplatně použit i na hledání chyb ve Firefoxu. Tentokrát poprvé je však tento nástroj svěřen přímo do rukou vývojářů. Od 6. března bylo nalezeno 327 chyb.

→Luboš Doležel ⋆ *11.8.2006*

Instalujeme a používáme phpMyAdmin

PhpMyAdmin je pravděpodobně jedním z nejpoužívanějších open source nástrojů psaných v PHP, používá se ke správě MySQL databází. Pokud nejste dosud obeznámeni s instalací a používáním tohoto software, dvoustránkový návod může být dobrým odrazovým můstkem.

→Luboš Doležel ⋆ *11.8.2006*

Mylo – přenosný handheld s Linuxem

LinuxDevices píše o přenosném handheldu Sony Mylo s podporou WiFi, který se bude trochu podobat Nokii 770. Toto zařízení bude poháněno Linuxem a grafické rozhraní bude založeno na Qtopia. Jeho cena se bude pohybovat okolo 350 dolarů.

→Luboš Doležel ⋆ *11.8.2006*

openSUSE slaví narozeniny

Uplynul rok od vzniku projektu openSUSE, což je celosvětový komunitní program sponzorovaný firmou Novell. Za cíl si klade poskytnout jednoduchý a bezplatný přístup k distribuci SUSE Linux a nabízí linuxovým vývojářům a nadšencům vše, co potřebují pro práci s Linuxem. Více informací se dozvíte na www.suseportal.cz.

→Jiří Větvička ⋆ *11.8.2006*

AMD + ATI = opensource ovladač?

Na InfoWorlde sa objavil článok o ďalšom postupe AMD po akvizícii ATI. Plný klasických poznámkach o posilnení voči konkurencii apod. Zaujímavý je však posledný odstavec. AMD uvažuje o vydaní driverov pre ATI grafické karty ako opensource!

→White Raven ⋆ *11.8.2006*

Vážný bezpečnostní problém Ruby on Rails

Ruby on Rails má velký bezpečnostní problém. Tvůrci okamžitě vydali opravnou verzi 1.1.5, kterou označují jako „povinný upgrade“. Chyba je prý tak závažná, že nezveřejnili žádné podrobnosti.

→Robert Krátký ⋆ *11.8.2006*

Balíčky ntfs-3g pro distribuci Ubuntu Dapper

Zak B. Elep potřeboval jednoho dne nainstalovat do svého Ubuntu podporu pro čtení a zápis NTFS. Za nejfunkčnější řešení vybral ntfs-3g. Jak už to tak bývá, instalace ze zdrojových kódů do binární distribuce je občas velmi „veselá“. Co všechno ho potkalo si můžete přečíst v jeho blogu. Důležité je, že pro nás ostatní vytvořil balíčky pro Ubuntu, abychom měli instalaci snazší.

→Jiří „Geo“ Lužnický ⋆ *11.8.2006*

Ovladače grafiky od Intelu již v Ubuntu Edgy

Používáte Ubuntu Edgy a chtěli jste vyzkoušet nové ovladače grafiky od Intelu [zprávička], ale nechtělo se vám je kompilovat? Nyní nemusíte. Dnes se nově objevily v oficiálních repozitářích.

→Jiří „Geo“ Lužnický ⋆ *11.8.2006*

Jaké je přednášet na počítačové konferenci?

Říkali jste si někdy, jak zábavné musí být přednášet na akcích jako Install Fest, Open Weekend a podobných. Kevin Shockey si řekl totéž o OSCONu. V jeho článku na OnLamp.com se dozvíte nejen o některých stinných stránek takového rozhodnutí, ale dostanete také několik dobrých rad.

→Jiří „Geo“ Lužnický ⋆ *11.8.2006*

Tajemství Gentoo Portage

Ostřílení uživatelé Gentoo jistě mají práci s Portage v malíčku, ale pro ostatní mohou být věci jako eix, ccache nebo distcc novinkou. Právě o nich a dalších si můžete přečíst v krátkém článku.

→Luboš Doležel ⋆ *12.8.2006*

Xara Xtreme 0.7

Xara Xtreme je velice kvalitní grafický program, který zvládá zároveň práci s vektory i s bitmapami. O otevření jejích zdrojových kódů už tady řeč byla. Po necelých pěti měsících byla v pátek konečně vypuštěna očekávaná verze 0.7, která přináší opět celou řadu změn, a jelikož se linuxová Xara funkčností

už hodně blíží své okenní sestře, dochází zároveň k přejmenování projektu z Xara LX zpět na Xara Xtreme. Už dnes se s Xarou na Linuxu dá docela slušně pracovat a při tomto tempu vývoje lze očekávat finální verzi 1.0 ještě letos.
→Martin Kocourek ⋆ *13.8.2006*

Debian/Ubuntu repozitář starých her pro DosBox

Na adrese dosboxed-games.sandbox.cz/ vzniká repozitář balíčků pro Debian s Abandonware PC hrami, které běží v DosBoxu. Hry by mělo být možno spustit okamžitě po instalaci bez nutnosti editovat konfigurační soubory DosBoxu. Nacházejí se zde kromě oficiálně uvolněných titulů i hry, které nemají vyjasněné licenční podmínky, takže nesplňují přísná kritéria Debianu a Liberated Games, ale jejichž autoři je již oficiálně nedistribuují a není možno je zakoupit.
→Vit Hrachovy ⋆ *14.8.2006*

Automatizované zálohování webu a MySQL databáze

O automatizovaném inkrementálním zálohování webu a MySQL databáze si můžete přečíst na nixCraftu. Vystačíte si s cronem a nástroji tar a mysqldump.
→Luboš Doležel ⋆ *14.8.2006*

Jednoduché stahování videí z YouTube

V posledních měsících se video-hostingy jako YouTube stávají populárními, ale ne všem vyhovuje přehrávání přes Flash ve stránce. Youtube-dl je skript v Pythonu, se kterým video z YouTube jednoduše stáhnete na svůj disk.
→Luboš Doležel ⋆ *14.8.2006*

HP bude poskytovat podporu pro Debian

Hewlett-Packard začne pro Debian nabízet možnost placené technické podpory. Debian se tak postaví mezi distribuce jako Red Hat Enterprise Linux nebo Novell Suse Linux Enterprise Server.
→Luboš Doležel ⋆ *14.8.2006*

Glade 3.0

Vyšel Glade 3.0, což je program pro návrh GUI pod Gnome a GTK+ . Došlo nejen k pročištění kódu, ale například i k integraci DevHelpu. Přibyla možnost tvorby vlastních menu a toolbarů a mnoho jiných věcí. Celý seznam změn najdeme na glade.gnome.org.
→Jiří „Geo“ Lužnický ⋆ *14.8.2006*

Linux From Scratch 6.2

„Distribuce“ Linux From Scratch dospěla do verze 6.2. Ta obsahuje například jádro ve verzi 2.6.16.26, GCC 4.0.3 a glibc 2.3.6. Dále se zapracovalo na manuálu, aby byl aktuální, přesnější a jasnější. Manuál je k dispozici v online-verzi i ke stažení. Live-CD vhodné pro instalaci najdete na ftp.osuosl.org (ISO).
→Jiří „Geo“ Lužnický ⋆ *14.8.2006*

Společná konference IBM a OSS Alliance

Organizace OSS Alliance a IBM vás zvou na konferenci s názvem „Zapřáhněte tučňáky do skutečné práce aneb Linux v reálném pracovním využití“.
→Filip Molčan ⋆ *15.8.2006*

Pozvánka na 11. setkání LVB 08/06

Jako každý třetí týden v měsíci, i tento pátek se v Brně koná setkání příznivců Linuxu a otevřených technologií. Tentokrát výjimečně na odlišném místě. Mapu a instrukce naleznete jako vždy na našem webu: linuxvbrne.org.
→Martin Sivák ⋆ *15.8.2006*

retty 1.0

Utilita retty 1.0 umožňuje dočasně přesunout program z libovolného terminálu na terminál aktuální. Stalo se vám někdy, že jste pustili program lokálně, někam jste odešli a pak potřebovali vzdáleně s tímto běžícím programem pracovat? Stačí se prostě zalogovat a na daný program spustit retty.
→Jan Šembera ⋆ *15.8.2006*

Ubuntu rozšiřuje pomoc novým uživatelům

Aby byla zaručena kvalita pomocí začínajícím uživatelům Ubuntu, byl vytvořen tým „NUN“ (NewUserNetwork). Členové tohoto týmu budou sledovat fórum, mailing list a IRC kanály, kde budou odpovídat na dotazy.
→Luboš Doležel ⋆ *15.8.2006*

gfiles.org: databáze aplikací pro GTK/Gnome

Na gfiles.org je k dispozici on-line katalog aplikací určených pro GTK/Gnome. Umožňuje nejen vyhledávání podle jména, ale i podle příznaků (tags), nebo také procházení po kategoriích.
→Jiří „Geo“ Lužnický ⋆ *15.8.2006*

Bitva o softwarové patenty v EU bude pokračovat

Bitva o softwarové patenty v EU se vrátí v zimě tohoto roku. Můžete se dočíst o rozdílech mezi předchozím a nadcházejícím soubojem a jak se bude lišit postavení odpůrců tohoto návrhu.
→Luboš Doležel ⋆ *15.8.2006*

Slackware Linux 11 se blíží

Vývoj Slackware Linuxu 11 se blíží ke konci. Jeden z vývojářů označil současný stav jako RC1. Aktuální seznam změn (pro platformu i386). Screenshoty na osdir.com. Neoficiální obraz CD naleznete na ftp.slackware.no.

→Jiří „Geo“ Lužnický ⋆ *15.8.2006*

Google Code Jam, začátek kvalifikace

Myslíte si, že patříte mezi nejlepší programátory na světě? Dokažte to. Příležitost máte v akci Google Code Jam. Včera začala kvalifikace, která potrvá do 14. 9. Tisíc nejlepších se pak zúčastní třech eliminačních kol. Vítězové si odnesou zajímavé ceny a slušný balík peněz.

→Jiří „Geo“ Lužnický ⋆ *15.8.2006*

První část zdrojového kódu Javy v říjnu

Sun Microsystems v květnu slíbil, že postupně uvolní zdrojový kód Javy pod open source licencí. Nyní společnost oznámila, že první části kódu budou uvolněny už v říjnu.

→Luboš Doležel ⋆ *15.8.2006*

DDM na Praze 9 hledá lektora pro linuxové kroužky

Dům dětí a mládeže hledá lektora pro kroužky „PC Linux“ a „PC Webaři“. Více informací na ddm-praha9.cz/volnamista.

→Redakce ⋆ *15.8.2006*

Stručný úvod do systému Linux

Je pro vás zdejší učebnice Linuxu příliš chudá a nepřehledná? Přecházíte z jiného OS a máte problém se v něm zorientovat? Zkuste si pročíst Úvod do systému Linux na serveru wraith.iglu.cz, který píše Petr Mach a vulgo Wraith.

→Aleš Kapica ⋆ *16.8.2006*

Glucose – OpenGL akcelerace pro X.Org

Zack Rusin včera přidal do Git stromu X.org novou akcelerační architekturu Glucose založenou kompletně na OpenGL. Glocuse využívá stejný základ kódu jako XGL a spolu s AIGLX poskytuje standardnímu X.Org serveru stejné možnosti jaké má Xglx server (veškerá 2D grafika je akcelerována prostřednictvím OpenGL). Více viz blog Zacka Rusina a xorg mailing-list.

→Michal Křenek ⋆ *16.8.2006*

Bazaar 0.9

Před několika dny vyšla verze 0.9 distribuovaného SCM systému Bazaar (došlo k přejmenování z Bazaar-NG zpět na Bazaar). Hlavní novinkou je především obrovské zvýšení rychlosti (u některých operací až šestinásobné!), výrazné zlepšení operací přes klasické FTP a opravení mnoha chyb.

→Michal Křenek ⋆ *16.8.2006*

Český seznam rozšíření pro Firefox podle činnosti

Na serveru Czilla.cz najdete překlad článku ze serveru eConsultant. Ten roztřídil více jak 100 rozšíření pro Firefox do kategorií, takže si lze velmi snadno najít to správné podle funkce, kterou od něj očekáváte.

→Jiří „Geo“ Lužnický ⋆ *16.8.2006*

Nový instalátor v Debian Etch

Debian Etch se v třetí betaverzi dočkal nového instalátoru. Ten obsahuje volitelnou instalaci v GUI a podporu šifrovaných oddílů.

→Luboš Doležel ⋆ *16.8.2006*

Gnutella klient se zvláštní licencí

GPU je jedním z klientů sítě Gnutella. Jeho zvláštností je jeho upravená licence GPL, která je rozšířená o zákaz vojenského použití software. O důvodech tohoto zákazu se píše na NewsForge.

→Luboš Doležel ⋆ *16.8.2006*

NetBeans 6 Milestone 2

NetBeans 6.0 Milestone 2. V nové verzi přibyla plovoucí okna a fullscreen mód (video ukázka), byl vylepšen debugger a do Profileru byl integrován JMeter. Oznámení o vydání a celý seznam změn najdete na wiki.netbeans.info.

→Jiří „Geo“ Lužnický ⋆ *16.8.2006*

Madriva Linux 2007 Beta 2

Vyšel Madriva Linux 2007 Beta 2. V nové verzi je obsaženo například nejnovější jádro (2.6.17.8), Gnome 2.16b2 nebo KDE 3.5.4. Byl změněn výchozí vzhled (zatím jen v Gnome). Distribuce je stále dostupná ve čtyřech verzích (kombinace Gnome/KDE a i586/x86_64). Oznámení o vydání najdete na mandriva.com.

→Jiří „Geo“ Lužnický ⋆ *16.8.2006*

Open source knihovna pro Windows Media od Real

Společnost Real oznámila záměr vytvořit open source knihovnu, která umožní přehrávání Windows Media na Linuxu, a to bez podpory DRM. Výsledný kód bude hned použit v přehrávači Helix.

→Luboš Doležel ⋆ *16.8.2006*

VMware Workstation 5.5.2, VMware Player 1.0.2

Ve velmi krátkám rozestupu vydala společnost VMware nové verze těchto produktů: VMware Workstation 5.5.2 (placený), WMware Player 1.0.2 (zdarma) a VMware Server 1.0.1 (zdarma). U prvních dvou jde o rozšíření podporovaných OS, u VMware Serveru je opraven problém s výkonem na 64bit Windows platformě. Při této příležitosti bych upozornil i na dokument na virtualization.info (PDF), kde jsou porovnány vlastnosti produktů Workstation a Server.

→vlho ⋆ *16.8.2006*

Slackware 11.0

Po posledních aktualizacích Slackware current, se již slackware-version přepisuje na 11.0 a mail od Patricka vítá do nové verze.

→Aldagautr ⋆ *17.8.2006*

Oraschemadoc 0.28

Generátor dokumentace schémat Oracle databází Oraschemadoc dospěl do verze 0.28. Kromě nových briliantních vlastností jsou připraveny zkompilované balíky i pro Linux, takže se uživatel vyhne ne zcela triviální instalaci potřebných pythonovských modulů?

→Petr Vaněk ⋆ *17.8.2006*

Testování použitelnosti v prostředí uživatele

Na AVC se konečně objevil záznam ze setkání Czech SIGCHI na téma „Testování použitelnosti v prostředí uživatele“.

→Martin Beránek ⋆ *17.8.2006*

Přes 20 000 studentů v Indianě používá Linux

Na základě státního projektu v americkém státě Indiana přešly pracovní stanice využívané více než 20 000 studenty na Linux. Z celkového počtu miliónu studentů se jedná jen o malý zlomek, ale i tak je to úspěch. Důvodem pro nasazení jsou nízké pořizovací náklady.

→Luboš Doležel ⋆ *17.8.2006*

Začínáme s OpenOffice.org Basicem

OpenOffice.org Basic je skriptovací jazyk poměrně podobný Visual Basicu. Seznámí vás s ním článek od Linux.com, ve kterém si vytvoříte skript pro vyhledávání výrazů ve slovníku nebo Wikipedii.

→Luboš Doležel ⋆ *17.8.2006*

Linuxové telefony ne tak svobodné

Mnoho z nás si přeje mobilní telefon poháněný Linuxem. Tom Sanders na LinuxWorldu upozornil, že od těchto zařízení nemůžeme očekávat takovou svobodu používání, jakou nám dává Linux na PC. Něco takového by se totiž nemuselo líbit regulačním orgánům.

→Luboš Doležel ⋆ *17.8.2006*

Reportáže a video z výstavy LinuxWorld

Už třetí den v San Francisku probíhá konference a výstava LinuxWorld. Na serveru Linux.com si můžete nejen přečíst reportáž z prvního a třetího dne, ale hlavně shlédnout pěti minutové video, a na chvíli získat pocit, jako byste tam opravdu byli.

→Jiří „Geo“ Lužnický ⋆ *17.8.2006*

Perl 5.9.4

Vyšel Perl 5.9.4. Oproti předchozí verzi byla například vylepšena podpora regexů a backportováno několik funkcí z verze 6 (například DOES()). Celý seznam změn. Oznámení na serveru OnLamp.com.

→Jiří „Geo“ Lužnický ⋆ *17.8.2006*

Bug v apt-get moo

Některá hlášení chyb mohou být opravdu veselá. Například doesnt look like a cow, které srovnává apt-get moo v Ubuntu a Gentoo. Kráva v Gentoo prý vypadá lépe. Autor bugreportu a několik diskutujících proto navrhli různé nové varianty.

→Jiří „Geo“ Lužnický ⋆ *17.8.2006*

Manuál k Blenderu v češtině

Ve wiki portálu kyberpunk.org vzniká neoficiální český překlad manuálu k Blenderu, což je open-source software pro modelování a vykreslování 3D počítačové grafiky. Originál manuálu najdete na mediawiki.blender.org, kam bude později přesunut i český překlad. Pokud máte chuť, můžete s překladem pomoci i vy.

→Jiří Větvička ⋆ *18.8.2006*

Kontrolujeme bezpečnost LAMPu s Nikto

Trápí vás zabezpečení vašeho webserveru s PHP a MySQL? Nikto je Perl skript, který může odhalit některé bezpečnostní problémy na vašem serveru, zkontroluje totiž nejen konfigurační soubory.

→Luboš Doležel ⋆ *18.8.2006*

Píšeme si poznámky s Tomboy

Tomboy je jednoduchá aplikace pro zápis poznámek v prostředí GNOME. Pokud vám něco takového schází, určitě si přečtěte recenzi na tento program.

→Luboš Doležel ⋆ *18.8.2006*

XenSource začne prodávat XenEnterprise

XenSource začne příští týden prodávat virtualizační software XenEnterprise, který má být přímým konkurentem VMware. Jeho použití má být snadné, má podporovat i běh neupravených Windows, a to díky podpoře virtualizace na procesorech Intel a AMD.

→Luboš Doležel ⋆ *18.8.2006*

Co je AIGLX, XGL, DRI, MesaGL a další

AIGLX, XGL, DRI, MesaGL a další. Máte v tom taky trochu nejasno. Článek na FreeSoftwareMagazine vám pomůže udělat si v těchto pojmech pořádek. Dozvíte se, o co se tyto věci starají, co potřebujete pro jejich funkci, a kdo je používá.

→Jiří „Geo“ Lužnický ⋆ *18.8.2006*

Deset tipů pro rychlejší práci v NetBeans

Rozhodli jste se používat prostředí NetBeans, ale nejde vám v něm práce od ruky tak jako dřív. Několik zajímavých tipů najdete v blogu 10 Tips For Coding In NetBeans. Většina tipů se týká věcí, které nemusíte dělat, ale udělají je NetBeans za vás.

→Jiří „Geo“ Lužnický ⋆ *18.8.2006*

GTK+ 2.10.2

Vyšlo GTK+ 2.10.2. Jako obvykle byla opravena spousta chyb (ačkoliv jedna z nejvážnějších stále zůstává #169870). Nezajímavější změny se týkají GtkFileChooser a GtkRecenManager.

→Jiří „Geo“ Lužnický ⋆ *18.8.2006*

VMware Server a linkované klony

Nedávno tu proběhla diskuze, že linkované klony umí jen VMware WorkStation. Pravdou je, že to jde i na VMware Serveru, což naznačuje tento postup. Uvedené se opírá o funkci Independent Disk. Další využití ukazuje článek o spouštění z CD/DVD. Přestože je použitý nástroj VMware Virtual Machine Importer zatím jen pro platformu Windows, dá se to udělat i na Linux platformě, jen ne tak „přátelsky“.

→vlho ⋆ *18.8.2006*

Ati-drivers verze 8.28.8

Vyšla nová verze proprietárních ovladačů karet Ati. Přináší nově podporu pro karty Radeon Xpress 1200 (1250, 1300) IGP. Stahovat můžete přímo od ati nebo samozřejmě počkat na distribuční balíček.

→Radek Dvořák ⋆ *19.8.2006*

První verze KDE4: 'Krash'

Projekt KDE oficiálně zveřejnil první vývojářskou verzi nadcházejícího KDE4 pod kódovým označením „Krash“. K dispozici jsou také RPM balíčky pro SUSE Linux.

→Jindřich Pozlovský ⋆ *19.8.2006*

První setkání JUG (Java User Group)

12. září proběhne na ČVUT první setkání JUGu (Java User Group). Na setkání promluví například viceprezidentka SUNu Laurie Tolson o opensourcování Javy a jiných aktuálních tématech. Prezentace bude nahrávat AVC SH a později budou dostupné ke stažení.

→Jiří „Geo“ Lužnický ⋆ *21.8.2006*

Gaim 2.0beta3.1

Po zmene protokolu MSN a následnom padaní IM klienta Gaim vývojári včera urýchlene vydali opravnú verziu 2.0beta3.1, ktorá okrem tohto problému rieši aj niekoľko ďalších chýb. Sťahujte zo SourceForge.

→Andrej Krivulčík ⋆ *21.8.2006*

Přijde MS Office pro Linux?

Šéf OSDL Stuart Cohen si myslí, že Microsoft během následujících let nabídne verzi balíku Office pro Linux, aby zastavil vzestup Sun Microsystems a OpenOffice.org. Více na LinuxInsider.

→Daniel Kvasnička ml. ⋆ *21.8.2006*

W3C aktualizovalo specifikace XML

eWeek informuje, že konsorcium W3C aktualizovalo specifikace XML 1.0 a 1.1. Z oficiálních míst také přišla informace, že ke konci roku 2006 se dočkáme nových standardů pro XML Query 1.0 a XSLT 2.0.

→Daniel Kvasnička ml. ⋆ *21.8.2006*

Bug Ubuntu číslo 1

Nová kritická chyba byla zveřejněna na launchpad.net. Jedná se o bug „Microsoft má majoritní podíl na trhu“ a Ubuntu byl navržen k jeho opravě.

→Lukas ⋆ *21.8.2006*

Skončila konference LinuxWorld

V pátek skončila konference/výstava LinuxWorld 2006 [zprávička]. Velmi obsáhlé shrnutí tohom co se dělo od prvního dne, si můžete přečíst na serveru OnLamp.com. Na serveru Newsforge pak ještě najdete reportáž z posledního dne.

→Jiří „Geo“ Lužnický ⋆ *21.8.2006*

Slackware Linux 11 RC 2

Aktuální stav vývoje Slackware Linuxu 11 byl označen jako RC 2. K dalším změnám by mělo už docházet pouze v případě nalezení chyb. Neoficiální ISO obrazy jsou stále k dispozici na ftp.slackware.no.

→Jiří „Geo“ Lužnický ⋆ *21.8.2006*

Připomínkování výukových textů k open source

MIČR oznámilo, že je možné připomínkovat texty k výukovým kurzům základů práce s počítačem a internetem, které na open source softwaru pro MIČR zajišťuje OSS Alliance.

→Filip Jirsák ⋆ *22.8.2006*

Skype for Linux 1.3.0.37 BETA

Vyšla další betaverze Skype pro Linux, které přinaší především opravy chyb a přidává přepínač pro povolení zobrazování stavu na webu.

→Filip Bartmann ⋆ *22.8.2006*

NetBSD 3.1 RC1

Vyšlo NetBSD 3.1 RC1. Zlepšuje zejména stabilitu LFS (souborový systém) a přidává podporu pro Xen3. Více detailů lze nalézt v oficiálním oznámení o vydání. Novou verzi lze získat například pomocí CVS nebo jako ISO image. Nicméně do čtrnácti dnů má vyjít RC2.

→Jiří „Geo“ Lužnický ⋆ *22.8.2006*

Byl oživen projekt podcastingového klienta Monopod

Po roce byl oživen vývoj programu Monopod, což je jednoduchý klient pro podcasting. Je určený „pro lidi, kteří si chtějí vybrat pár kanálů a po chvíli mít připravené MP3 na hardisku.“ Nová verze nepřináší nic nového, pouze opravuje chyby.

→Jiří „Geo“ Lužnický ⋆ *22.8.2006*

Xgl na Ubuntu opravdu snadno a rychle

Na help.ubuntu.com se nedávno objevil přehledný a úplný návod, jak Xgl zprovoznit, aniž by vám hrozilo, že si ze systému uděláte smetiště. Instalace a nastavení jsou popsány pro KDE/Gnome/XFce/ a graf. karty nVIDIA/ATI/Intel.

→Jiří „Geo“ Lužnický ⋆ *22.8.2006*

XGL v Mandriva 2007

Programátoři se začali činit také v Mandrivě. V nastupující verzi pro rok 2007, která by tu měla být co nevidět, bude implementováno akcelerované grafické XGL prostředí.

→David ⋆ *23.8.2006*

Amarok 1.4.2

Amarok 1.4.2 (druhá opravná verze série Fast Forward) přidává podporu streamování pomocí protokolu DAAP (takže je možné se připojit např. k iTunes). Kromě toho přibyla základní podpora dynamických kolekcí (výsledek Summer of Code).

→Robert Krátký ⋆ *23.8.2006*

Kickoff: nové KDE 'Start menu' v openSUSE 10.2

openSUSE 10.2 bude mít v KDE nové 'Start menu' – Kickoff. Bude obsahovat například vyhledávání. Ukázku najdete na stránce Stephana Binnera.

→Robert Krátký ⋆ *23.8.2006*

SLAX 5.1.8rc1 – podpora zápisu na NTFS (NTFS-3g)

SLAX 5.1.8rc1 obsahuje – mimo jiné – podporu pro bezproblémový zápis na NTFS oddíly (díky začlenění NTFS-3g). Celý seznam změn najdete na slax.org/changelog.php.

→Robert Krátký ⋆ *23.8.2006*

Bonfire byl přejmenován na Brasero

Program Bonfire (vypalování CD/DVD) se od verze 0.4.2 jmenuje Brasero. Bonfire je totiž registrovaná obchodní známka. Nová verze navíc přináší opravy několika chyb, které způsobovaly pád programu.

→Jiří „Geo“ Lužnický ⋆ *23.8.2006*

Chorvatská státní správa podporuje OS

Chorvatská státní správa přijala open source politiku pro používání a vývoj software. Mezi pokyny je například, že instituce jsou povinny vybírat open source řešení namísto uzavřených alternativ, dále podpora otevřených formátů i mimo samotné instituce nebo zrovnoprávnění otevřených a uzavřených řešení ve výuce. Informoval server Newsforge.com.

→Jiří „Geo“ Lužnický ⋆ *23.8.2006*

Rozhovor s vývojářem NetBeans C/C++ Pack

Na serveru NetBeans.org vyšel zajímavý rozhovor s Vladimirem Voskresenskym. Ten je šéfem týmu vývojářů C/C++ Packu. Rozhovor se netýká pouze samotného Packu, ale poměrně dost se čtenář dozví i o některých nových vychytávkách v API.

→Jiří „Geo“ Lužnický ⋆ *23.8.2006*

Na počátku byl SHELL

Vesmírná sonda WMAP, která scanuje vesmír z doby velkého třesku, objevila na svých snímcích vyjádřena písmena „SH“. Nezbývá, než se jen domnívat, že u zrodu našeho vesmíru stál SHELL :-).

→Pavel Škoda ⋆ *24.8.2006*

Wikipedia v RESPEKTu

V aktuálním čísle časopisu RESPEKT je k přečtení zajímavý článek o Wikipedii doplněný o rozhovor, ve kterém David Gerard (jedna z vůdčích postav anglické verze projektu) vysvětluje, proč Wikipedia na své stránky nepouští reklamu.

→Daniel Kvasnička ml. ⋆ *24.8.2006*

Wine 0.9.20

Vyšlo Wine 0.9.20. Přináší zlepšení podpory OpenGL a IDL kompilátoru. Spousta textů je nyní lokalizovatelná místo hardcoded angličtiny. Opravuje pro spoustu různých chyb.

→David Watzke ⋆ *24.8.2006*

Nvidia ovladače s podporou X.Org 7.1

Nově vydané binární Nvidia ovladače 1.0-8774 konečně podporují X.Org 7.1! Dále má být zlepšena interakce s novějšími linuxovými jádry. Vše podstatné na stránkách Nvidie.

→Radomír Fojtík ⋆ *24.8.2006*

wxDVDShrink

Na Sourceforge.net je moznost stáhnout první prealpha verzi programu wxDVDShrink. Který jak již název napovídá slouží ke kopírování DVD9 na DVD5. Tak stahujte, kompilujte, testujte a posílejte náměty na zlepšení.

→Daniel Kozák ⋆ *24.8.2006*

l10n.kde.sk v plnom švungu

Po týždeň starej výzve možno pozorovať nevídanú aktivitu v lokalizovaní KDE. Už teraz sa dá povedať, že KDE 3.5.5 bude jedno z „najslovenskejších" vôbec :-) Pomoc je stále vítaná (aktuálny stav)

→Jozef Říha ⋆ *26.8.2006*

Apple otevře části WebObjects

Po nedávném flirtu s open source Apple ve čtvrtek odkryl plány na otevření většiny zdrojových kódů javovského aplikačního serveru WebObjects. Informuje AppleInsider.

→Daniel Kvasnička ml. ⋆ *27.8.2006*

Upstart – další náhrada initu

Zajímavý blogpost (Upstart in Universe) pro zájemce o problematiku (nejen pythonovských, viz Init skripty v Pythonu (nesmělé krůčky)) „next-generation-init" skriptů/démonů.

→Basque Border ⋆ *27.8.2006*

KPhotoAlbum – soutěž o nový splashcreen

KPhotoAlbum (dříva KimDaBa), program pro správu sbírek fotografií, vyhlásil soutěž o nový splashscreen. Odměnou pro výherce bude 100 USD.

→Robert Krátký ⋆ *28.8.2006*

Intel pomáhá vietnamským komunistům s open source

Vietnamská komunistická strana se rozhodla přejít na open source. A Intel jim s tím pomůže – připraví tzv. OpenLab pro testování a vývoj open source softwaru.

→Robert Krátký ⋆ *28.8.2006*

Debian Etch Beta 3 – recenze

All about Linux nabízí své dojmy z instalace a prvních minut používání Debian Etch Beta 3 (příští stabilní Debian, očekávané vydání koncem roku 2006).

→Robert Krátký ⋆ *28.8.2006*

Obnova smazaných souborů

Nechtěně smazané soubory nemusejí být trvale ztracené. Můžete se pokusit o záchranu některých souborů pomocí TestDisk a PhotoRec a doufat, že budete mít štěstí.

→Luboš Doležel ⋆ *28.8.2006*

Píšeme text s minimalistickými nástroji

Píšete mnoho textu, ale chcete se zbavit klasického textového procesoru? Článek na Linux.com navrhuje používat Vim a na další zpracování textu txt2tags.

→Luboš Doležel ⋆ *28.8.2006*

Yukon – nástroj pro nahrávání videa z her

Hráčům na Linuxu dlouho scházel nástroj na nahrávání videa ze hry. Tuto mezeru zaplnil Yukon, který má dobrý potenciál stát se Frapsem/GameCamem pro Linux.

→Luboš Doležel ⋆ *28.8.2006*

České TV programy do XMLTV

Chcete-li stahovat české TV programy (ve formátu XMLTV), můžete využít nový program tv_grab_cz (napsaný v Pythonu). Program získává data ze serveru 365dni.sms.cz. Vygenerovaný XML soubor je zatím ještě nutné prohnat programem tv_sort z balíku xmltv. Více viz blogpost.

→Michal Křenek ⋆ *28.8.2006*

Slackware Linux 11 RC3

Po několika málo dnech od vydání RC2 je zde Slackware Linux 11 RC3. Všechny „neodvolatelné" změny už byly udělány. Bugů zbývá jen pár. Ještě se možná dočkáme dalších změn a RC4, ale dle seznamu změn už jsme blízko finální verze. Seznam mirrorů s vývojovou verzí najdete na abnormalpenguin.com.

→Jiří „Geo" Lužnický ⋆ *28.8.2006*

Studie efektivity emailových konferencí

Rozsáhlá studie na serveru OnLamp.com se zabývá efektivitou mailových konferencí. První díl popisuje motivaci a cíle autorů, dále pak postup „měření". Druhý díl jistě potěší milovníky grafů a různých

vztahů. Poslední díl obsahuje zejména hodnocení užitečnosti zpráv a samozřejmě závěry měření.
→Jiří „Geo“ Lužnický ⋆ *28.8.2006*

Mediastation X-76: hudební workstation

Mediastation firmy Lionstracs je po Korgu Oasys další hudební workstation poháněná Linuxem. Vyrábí se v několika variantách a mimo mnoha nativních aplikací si poradí i s VST.
→Ctirad Fečtr ⋆ *29.8.2006*

Co brání GIMPu sesadit Photoshop z trůnu?

The Linux Advocate konstatuje, že GIMP zatím nemá na to, aby sesadil Adobe Photoshop z trůnu grafických editorů a ve stručnosti popisuje, co považuje za největší nevýhody GIMPu. Mimo jiné je to i samotný název programu. GIMP vs. Photoshop – What still needs to be done?
→Daniel Kvasnička ml. ⋆ *29.8.2006*

10 nejčastějších neporozumění licenci GPL

GNU GPL je beze sporů jednou z nejčastěji užívaných open source licencí, ale i přesto jí ne všichni dobře chápou. Na IT Manager's Journal si můžete přečíst 10 nejčastějších chyb při porozumění této licenci s uvedením na pravou míru.
→Luboš Doležel ⋆ *29.8.2006*

Projekt Linux Libertine Open Fonts

Projekt Linux Libertine Open Fonts si klade za cíl vytvořit svobodné varianty některých často užívaných fontů, které spadají do skupiny „corefonts“. Už nyní je k dispozici obdoba Times New Roman.
→Luboš Doležel ⋆ *29.8.2006*

Dokáže Linux zachránit Palm OS?

Už jsou tomu 2 roky, co vyšla poslední aktualizace Palm OS. Příští rok by měla vyjít nová verze, která se bude hodně lišit – bude nasazena nová platforma Access Linux Platform. Dokáže vzkřísit Palm OS?
→Luboš Doležel ⋆ *29.8.2006*

Aircrak-ng 0.6.1

Především vyšel program Aircrak-ng 0.6.1. Byly vyřešeny některé problémy s přepínáním do „Monitor“ módu. Nově je také možné díky přidané podpoře pro shell ash spustit aircrack-ng na OpenWRT a Zaurus.
→Jiří „Geo“ Lužnický ⋆ *29.8.2006*

Vývojáři si žádají testování NB Profileru

V novém NetBeans Profileru byla přidána funkce „HeapWalker“. Vývoj této komponenty je teprve na začátku, proto vývojáři čekají na reakce uživatelů. Na profiler.netbeans.org si proto můžete vyzkoušet jednoduchý preview.
→Jiří „Geo“ Lužnický ⋆ *29.8.2006*

QCM platinovým partnerem Mandrivy

Firma QCM se stala platinovým partnerem Mandrivy. Jde o nejvyšší možnou formu spolupráce mezi Mandrivou a jejím partnerem. V rámci ní budou obě společnosti spolupracovat např. na korporátních produktech a jejich nasazení. Součástí dohody je i převod domény mandriva.cz do správy QCM. Více v tiskové zprávě.
→Ivan Bibr ⋆ *30.8.2006*

Firefox 2 beta2 rc2

Dnes byl vydán druhý release candidate druhé bety Firefoxu 2. Oproti 1.5.0.x byl vylepšen tabbed browsing, bylo vytvořeno nové výchozí téma a jednotný správce doplňků – pro správu témat a rozšíření – a mnoho dalšího. Stahujte na 2.0b2-candidates/rc2.
→Martin Sourada ⋆ *30.8.2006*

OpenOffice.org Basic: tvorba hyperlinků

Po předchozím článku [zprávička] vyšel další díl úvodu do jazyka OpenOffice.org Basic. Tentokrát se probírá vytváření hyperlinků a základní změny stylu textu.
→Luboš Doležel ⋆ *30.8.2006*

Převádíme videa z YouTube do MPEG-4

Odkazy na videa z YouTube se nacházejí po celém webu. Někoho však může omezovat formát, ve kterém jsou uloženy, a proto Manolis Tzanidakis uvolnil jednoduchý skript pro konverzi do MPEG-4. Může se vám hodit spolu se skriptem na stahování těchto videí.
→Luboš Doležel ⋆ *30.8.2006*

Interview s vývojáři Mindware Studios

Phoronix.com přináší interview se dvěma vývojáři společnosti Mindware Studios a tvůrci hry Coldwar, Patrikem Rakem a Tomášem Pluhaříkem. Baví se o problémech při vývoji a vydání této hry pro Linux a možnosti dalších her pro Linux.
→Luboš Doležel ⋆ *30.8.2006*

Google nabídne literární klasiku

Podle ekonomického serveru ČTK Google dnešním dnem začne nabízet bezplatné stahování a tištění literárních děl, která nejsou již chráněna autorskými právy. Google Book Search zatím dovoloval pouze díla prohledávat a číst on-line.
→Daniel Kvasnička ml. ⋆ *30.8.2006*

Mandriva Linux beta 3

Na mirrorech se pomalu objevuje Mandriva Linux 2007.0 beta 3. Stahujte, testujte, hlaste chyby.

→hajma ⋆ *30.8.2006*

Gentoo 2006.1

Včera byl oficiálně vydán nový profil Gentoo 2006.1. Nejpopulárnější architektury používají GCC 4.1.1, glibc 2.4 a baselayout 1.12.1. Dále, mimo jiné, přináší podprofily desktop a server.

→David Watzke ⋆ *31.8.2006*

Mono 1.1.17

Vyšlo Mono ve verzi 1.1.17. Mezi novinkami najdete např. nový runtime pro VB 2005 nebo IronPython 1.0 RC2. Poznámky o vydání, Download.

→Daniel Kvasnička ml. ⋆ *31.8.2006*

Změny v MySQL 5.1.12

MySQL ve verzi 5.1.12 vyloučí Berkley DB (BDB) engine. Prý se nejedná o krok související s odkoupení společnosti Sleepycat (vývojáři BDB) Oraclem, ale jde čistě o technické důvody. Ve verzi 5.1.12 se také objeví memcache plugin.

→Luboš Doležel ⋆ *31.8.2006*

Obavy o budoucnost NetBSD

Jeden ze zakladatelů NetBSD, Charles Hannum, se obává o budoucnost tohoto OS. NetBSD prý zaostává za ostatními OS a NetBSD Foundation, která nyní řídí celý projekt, údajně narušuje vývoj.

→Luboš Doležel ⋆ *31.8.2006*

OpenOffice.org Premium

Skupina nadšenců uvolnila OpenOffice.org Premium. Kromě samotného OpenOffice.org obsahuje navíc bonusy, jako např. šablony, kliparty nebo fonty. Podle slova „premium“ by se mohlo zdát, že balík je placený, ale není tomu tak.

→Luboš Doležel ⋆ *31.8.2006*

První část ze 100 tipů k NetBeans

Jeden z uživatelů NetBeans se rozhodl, že posbírá všechny tipy ohledně IDE, které se objevily v různých blozích. První část sbírky s názvem „100 IDE hacks“ si můžete přečíst přečíst na netbeans.org. Navíc jednu zajímavou vychytávku s hromadným nahrazováním přidává sám autor na blog.sun.com.

→Jiří „Geo“ Lužnický ⋆ *31.8.2006*

GUI (nejen) pro konfiguraci DHCP

Nedávno zde vyšel článek o DHCP. Pokud se bojíte příkazové řádky, nebo jste prostě líní ji používat, můžete zkusit program Gdhcpd, který vám s konfigurací pomůže. Gdhcpd je navíc součástí Gadmintools, které obsahují „klikátka“ například pro Sambu nebo Bind.

→Jiří „Geo“ Lužnický ⋆ *31.8.2006*

Wiki jako PIM

Server Linux.com prezentuje WiKi nejen jako nástroj pro sdílení informací a spolupráci na velkých projektech, ale i jako alternativu k PIM aplikacím. V článku jsou představeny tři malé wiki, které jsou velmi jednoduché na používání i rozchození.

→Jiří „Geo“ Lužnický ⋆ *31.8.2006*