Učebnice ABC/Linuxu

ze dne 19.10.2006

rozšířené vydání

Učebnice GNU/Linuxu

Každý začátek je těžký a obtížný, zvláště nemáte-li po ruce někoho zkušenějšího, kdo by vám radil a předával své zkušenosti. Operační systém GNU/Linux je velmi rozsáhlý a má místy velmi odlišné ovládání například oproti MS Windows. Proto je lepší přečíst si o něm knihu, která by vám prozradila jeho taje a zákoutí. Pak proniknete do jeho filozofie, naučíte se ovládat svůj počítač skutečně efektivně a rozšíříte své obzory. Našim cílem je poskytnout vám základní informace, které vám usnadní pochopení GNU/Linuxu.

Na trhu existuje spousta kvalitních knih na toto téma. V rubrice Recenze jich pár najdete, kompletní seznam je dostupný na linux.cz. Nicméně pokud teprve začínáte, nemusíte chtít investovat do koupě knihy, nemluvě o času stráveném na cestě do knihkupectví a zpět. Taková kniha přes své nesporné kvality může navíc časem zastarat nebo obsahovat chyby. Proto jsme se rozhodli usnadnit cestu ke GNU/Linuxu a připravili jsme tuto online učebnici Linuxu.

Tuto učebnici napsali a spravují čtenáři portálu www.abclinuxu.cz a je dostupná pod licencí GNU Free documentation license. Můžete si ji zdarma přečíst, stáhnout, vytisknout atd. K dispozici je i nepravidelně aktualizovaná PDF verze učebnice. "Odkaz na aktuální PDF verzi je zatím pouze na webu autora." Každý čtenář má možnost sám opravit text, chybu či překlep nebo doplnit čerstvé údaje. Nebojte se tuto možnost využít. Více o tomto projektu na jeho stránkách. Své názory nebo rady nám můžete sdělit v diskusním fóru.

Jak nám můžete pomoci:

- pište chybějící obsah nebo doplňte informace
- opravujte překlepy a pravopisné chyby, sjednocujte sloh
- doplňujte odkazy na zajímavé články a informace
- umístěte na své stránky odkazy na učebnici

Doufáme, že se vám bude učebnice líbit a že vám zpříjemní cestu k poznání GNU/Linuxu.

Obsah

- Na co se často ptáme: X Window systém
- Souborové systémy - často kladené otázky
- Dodatky k učebnici
- Čím v Linuxu nahradit aplikace Windows?

Stručný výtah

Linux, či přesněji GNU/Linux je moderní operační systém. Skládá se z jádra zvaného Linux, jehož autorem je Linus Torvalds a základních knihoven a nástrojů pocházejících převážně z projektu GNU, který založil Richard Stallman. GNU/Linux je především svobodný, sofistikovaný a univerzální systém. Můžete jej nasadit do embedded zařízení, na osobní počítače či servery, ale i na sálové počítače. Byl portován na širokou paletu architektur, od běžné x86 (osobní počítače 386 a vyšší), přes PowerPC (MacIntosh), Sparc (Sun Microsystems), Motorola 68k až třeba po MIPS.

Základ

GNU/Linux je svobodný operační systém unixového typu. Jeho jádro - Linux - bylo vytvořeno Linusem Torvaldsem za pomoci vývojářů z celého světa. Linux je vyvíjen pod licencí GNU General Public License, takže jeho zdrojový kód je volně k dispozici každému.

GNU/Linux vs. Linux

Na tomto místě je vhodné umístit vsuvku osvětlující poněkud kontroverzní názvosloví. Již poměrně dlouho předtím (1984), než Linus Torvalds začal (1991) s vývojem linuxového kernelu (jádro operačního systému), byl Richardem M. Stallmanem započat vývoj systému GNU (rekurzivní zkratka slov GNU's Not Unix = GNU není Unix). Cílem R. Stallmana bylo (a je) vytvořit operační systém, který nebude zatížen copyrightem. Za tím účelem byla vytvořena speciální licence, která doslova obrací copyright (také proto bylo pro vyjádření jejího principu zvoleno pojmenování Dona Hopkinse: copyleft). Nejstriktnější formou copyleftové licence, která je použita pro všechny zásadní programové části GNU, je GNU GPL (General Public License = Všeobecná veřejná licence). Velice stručně lze GNU GPL shrnout asi takto: předmět licence může být používán, kopírován, pozměňován a distribuován - naopak žádná jeho část nesmí být zatížena licencí nekompatibilní s GPL. O něco později byl GNU Project zastřešen FSF (Free Software Foundation = Nadace svobodného software).

V rámci aktivit GNU bylo vyvinuto mnoho programů, které postupně zcela plnohodnotně nahradily vitální komponenty dosavadních proprietárních unixových systémů. Jedinou věcí, která bolestně scházela, bylo jádro neboli kernel. Přestože vývoj originálního GNU kernelu stále pokračuje (možná jste zaslechli jméno HURD), zatím (a hlavně před dvanácti lety) není tak docela způsobilý k ostrému nasazení. Tato mezera byla zaplněna jádrem, které začal programovat L. Torvalds - Linuxem. Kombinací softwaru vzniklého pod hlavičkou GNU a kernelu Linux se zrodil kompletní operační systém schopný samostatného provozu bez jakékoliv součásti, jež by nevyhovovala GPL.

Z předchozího je tedy doufám zřetelně jasné, že Linux bez GNU a naopak je prakticky nepoužitelný. Korektní název stávajícího operačního systému by tedy měl znít GNU/Linux. GNU komunita v čele s R. Stallmanem je dost jednoznačná ve svém názoru a nazývání celého systému pouze zkráceným názvem Linux považuje za hrubou nepřesnost. Proti jejich argumentaci nelze samozřejmě mnoho namítat, avšak skutečnost je taková, že v praxi naprostá většina uživatelů používá jednodušší název Linux, ale myslí tím GNU/Linux, nikoliv pouze kernel.

Linus Torvalds

V roce 1991 začal finský student Linus Torvalds pracovat na projektu nového operačního systému pro počítače s procesorem Intel 386. Začal jej vytvářet jako semestrální práci založenou na operačním systému Minix profesora Andy Tanenbauma, což byl v té době de facto standardní unixový operační systém pro osobní použití. Linus se však prozíravě rozhodl podělit se o plody své práce i s ostatními a zveřejnil zdrojové kódy svého projektu na internetu pod licencí GNU GPL. Nový operační systém byl nazván po něm - Linux.

Zvolená licence měla jednu obrovskou výhodu - zajišťovala každému uživateli zdarma přístup ke zdrojovým kódům. V té době byli uživatelé především programátoři, takže tato výhoda byla pro ně veskrze praktická. Zároveň licence zajišťovala, že se již nikdy nezmění, takže se nemuseli obávat, že by jejich práci mohl někdo zneužít a vytvořit uzavřenou verzi Linuxu šířenou pod jinou licencí.

Linux tedy získal velkou pozornost mezi programátory, kteří dlouho očekávali takovýto projekt. Přidali se a začali zasílat opravy a vylepšení. Linux tak rychle zažil prudký vývoj a během několika málo let byl dokončen do podoby, kdy měl všechny základní služby. Od té doby prošel mnoha změnami a dospěl. Již od verze 2.0 může směle konkurovat operačním systémům nižší třídy (typu Windows 95/98, NT, 2000, různé varianty BSD), od verze 2.4 většině komerčních Unixů a verze 2.6 jej posunuje na dosah špičkových operačních systémů.

Co je to Linux

Linux ideově vychází z kořenu Unixu, jehož počátky sahají do sedmdesátých let. Přestože s původním Unixem nesdílí ani řádku zdrojového kódu, má stejné aplikační rozhraní i filozofii. Díky tomu je kompatibilní s ostatními Unixy na úrovní zdrojových kódů. Proto má i stejné aplikace a nástroje. Linux podporuje souběžnou práci více uživatelů, z nichž každý může spouštět libovolný počet programů. Linux byl upraven (portován) na spoustu jiných platforem, než je obvyklé PC. Najdete jej od kapesních počítačů s procesory Arm, přes Macintoshe, pracovní stanice od Sunu až po mainframy od IBM.

Aplikace

Mezi nejčastější mýty patří, že Linux se ovládá příkazovou řádkou a nemá grafické rozhraní. Opak je pravdou. O grafické rozhraní se stará standardní knihovna X Window System, v našem případě jeho implementace XFree86/X.org. K dispozici je spousta grafických prostředí, hlavními asi jsou KDE a Gnome, jejichž uživatelská přívětivost i grafická propracovanost patří ke špičce.

Počet aplikací pro Linux se počítá na desetitisíce. K dispozici je mimo jiné kvalitní prohlížeč Mozilla Firefox, komplexní kancelářský balík OpenOffice.org, na grafiku je výborný Gimp, multimédia si přehrajete v MPlayeru a tak bych mohl pokračovat. Viz náš přehled aplikací.

Podpora

Začátečníci se asi nejvíce bojí, že se nebudou mít z čeho Linux naučit, že jim nikdo nepomůže překonat nástrahy jeho odlišné filozofie, že jej nebudou umět ovládat. Zbytečně. Moderní distribuce přechod usnadňují. Nováčci si mohou v knihkupectvích vybírat mezi různými knížkami, které je naučí chápat i ovládat Linux. Na internetu je také spousta informací, například doporučuji přečíst si články, které jsme publikovali. Cenným zdrojem informací je i archiv diskusí a naše fulltextové vyhledávání. Nicméně nejdříve byste se asi měli naučit, jak a kde hledat linuxové informace.

Kde se vzal tučňák?

Sympaticky buclatý a dobrácký tučňák je nezaměnitelné logo Linuxu. Nebyl tu však odjakživa. Vlastně je s linuxovým jádrem spojován až od verze 2.0. Grafické zpracování má na svědomí Larry Ewing a původní idea pochází z "dílny" Linux Kernel Mailing Listu.

O tučňákovi řekl Linus Torvalds mimo jiné toto (u příležitosti oznámení verze 2.0 kernelu na Usenetu):

Some people have told me they don't think a fat penguin really embodies the grace of Linux, which just tells me they have never seen a angry penguin charging at them in excess of 100mph. They'd be a lot more careful about what they say if they had.

(Lidé mi říkali, že tlustý tučňák podle nich eleganci Linuxu zrovna nevystihuje. To je pro mne pouze důkazem, že nikdy neviděli rozzlobeného tučňáka, jak se na ně řítí rychlostí přes stošedesát kilometrů za hodinu, protože kdyby ho viděli, dávali by si o hodně větší pozor, co povídají.)

Distribuce

(z angl. "to distribute" = šířit, rozdat, rozdělit, roznést, rozšířit, rozprostřít)

Pro začátečníky může být obtížné pochopit, že GNU/Linux není jen jeden. Chtějí si vyzkoušet ten Linux, ale najednou se dozvídají, že si mají vybrat mezi Red Hatem, Mandrivou, SuSE či Debianem. U Windows je situace jednoduchá, dodává je jediná firma a tou je Microsoft. GNU/Linux může dodávat kdokoliv. A tak vznikly desítky či stovky profesionálních, ale i amatérských distribucí.

Nejlepší bude vysvětlit situaci na příkladě. V každém obchodě naleznete nejrůznější balené vody. Liší se značkou, obalem a složením. Přesto každá z nich obsahuje čistou vodu. U distribucí je situace podobná - všechny obsahují společný základ (jádro Linux, knihovny a nástroje od GNU), liší se ale balením (instalátor) a složením (výběr programů, unikátní úpravy). Pochopitelně každá distribuce má svou značku.

Možná se divíte, jak je možné, že distribucí je jako hub po dešti. Odpověď je prostá - velké množství distribucí vzniklo upravením jiné stávající distribuce. Mluvíme pak o něco-based distribucích, např. Debian-based. V podstatě všechny distribuce vycházejí buďto z Slackware, Debianu nebo Red Hatu. Je důležité mít toto na paměti, neboť Vám to může docela ulehčit život.

Distribuce můžeme členit podle nejrůznějších kritérií. Za některými stojí komerční firmy, které dodávají technickou podporu (Red Hat, Mandriva, Novell či TurboLinux), jiné jsou tvořeny komunitou (Debian, Fedora, OpenSUSE, Slackware, Gentoo). Některé jsou určeny na kancelářskou práci či domácí použití, jiné pro budování internetových serverů, další jsou zcela univerzální a některé jsou jednoúčelové (přehrávání DVD, síťové komponenty). Některé si můžete stáhnout z internetu zcela zdarma, za jiné zaplatíte tisíce korun a pro podniky existují verze stojící tisíce dolarů.

Seznam distribucí

V následujících odstavcích si můžete přečíst základní charakteristiku nejznámějších distribucí vhodných pro desktop. Pokud vás zajímá jejich popularita v našich zemích, přečtěte si výsledky poslední ankety. Kompletní seznam je k dispozici na distrowatch. Ale nenechejte se distrowatchem zmást, protože uvádí kromě distribucí GNU/Linuxu také distribuce BSD.

Arch Linux

[recenze: Arch Linux] Arch Linux je distribuce pro pokročilejší uživatele. Spojuje výhody Slackwaru (jednoduché init-skripty, čistý základní systém) s výhodami Gentoo (výborný balíčkovací manager pacman, možnost jednoduché tvorby balíčků ze zdrojáků - Arch Build System). Distribuce je optimalizována pro i686 a je velmi rychlá. Striktně se drží filosofie "rolling updates" (neexistují "stable" verze, uživatel si systém pravidelně aktualizuje z internetu na vždy nejnovější verze programů).

web získat

Debian

[recenze: Debian] Debian je jedna z nejstarších distribucí, založená Ianem Murdockem. (Jeho žena byla Debra - odtud název Debian). Jedná se o striktně open-source distribuci, vyvíjenou dobrovolníky, kteří mají dokonce vlastní ústavu. Debian je v současné době přeportován na 11 hardwarových architektur a nabízí veliký on-line repositář softwarových balíků (okolo 10 000).

web získat

Fedora

[recenze: Fedora Core 4] Fedora je distribuce sponzorovaná firmou Red Hat (a je také na Red Hatu založená). Je to však svobodná distribuce, vznikající podobně jako Debian. Fedora si klade velký důraz na bezpečnost a otevřenost.

web získat

Gentoo

[recenze: Gentoo] Pokud potřebujete 100% výkon a máte hodně času, volte Gentoo. Je diametrálně odlišné od všech ostatních - neinstaluje se ve formě binárních souborů, ale kompiluje se přímo na Vašem PC. Za cenu opravdu dlouhé instalace dostanete software plně optimalizovaný přesně na Váš konkrétní počítač.

web získat

Mandriva Linux

[recenze: Mandriva Linux 2006 CZ] Dříve Mandrakelinux, po sloučení společností Mandrake a Conectiva přejmenován do dnešní podoby. Distribuce používá balíčkovací systém RPM (stejně jako Red Hat / Fedora). Mandrivu často volí začátečníci, protože má funkčnost "out-of-the box" - tedy po asi 30 minutách téměř bezobslužné instalace máte plně funkční systém.

web získat

Red Hat

Jedna z nejstarších distribucí - Red Hat (červený klobouk). V dnešní době vysoce komerční, především díky Red Hat Enterprise Linuxu. V dávných dobách v Red Hatu vznikl dnes velmi rozšířený balíčkovací systém RPM. RH samotný se však na desktopech už příliš nepoužívá. Velká masa uživatelů přešla na distribuce na něm založené (hlavně Fedora).

web získat

Slackware

[recenze: Slackware 9.1] Slackware patří mezi starší distribuce a také by se dalo říct, že mezi distribuce pro pokročilejší. Po instalaci Vás "uvítá" pouze konzole, takže musíte udělat pěkný kus práce než dostanete alespoň X. Nicméně, někdo to má rád tvrdé. Při instalaci a upgradu softwaru v Slackware si musíme dávat pozor, protože balíčkovací systém Slacku nekontroluje závislosti.

web získat

SUSE (Novell)

[recenze openSUSE 10.0] SUSE byla původně samostatná distribuce, avšak později byla koupena firmou Novell. Je to distribuce velmi přívětivá i pro začátečníky. SUSE se specializuje na prodej krabicových verzí, avšak dá se stáhnout i zdarma z internetu. Novell podporuje komunitní otevřený vývoj distribuce ve formě projektu openSUSE (podobně jako Red Hat podporuje projekt Fedora).

web SUSE získat SUSE web openSUSE získat openSUSE

Ubuntu

[recenze Ubuntu 5.10] Ubuntu je distribuce vycházející z Debianu specializující se na desktop. Je volně dostupná (včetně možnosti bezplatného zaslání spustitelných a instalačních CD) a k dispozici je komunitní i profesionální podpora. Ubuntu je vhodné i pro začátečníky. Nové verze vycházejí každých šest měsíců a každá z nich je podporována vydáváním bezpečnostních aktualizací a oprav nejméně po dobu 18 měsíců. Kromě Ubuntu existuje i Kubuntu, která je postaveno okolo desktopového prostředí KDE (Ubuntu je postaveno okolo GNOME). Nejedná se však o další samostatnou distribuci, Kubuntu je součástí Ubuntu projektu (z Ubuntu je možné udělat Kubuntu a naopak).

web Ubuntu získat Ubuntu web Kubuntu získat Kubuntu

Aurox

Aurox je mladá linuxová distritribuce na bázi Fedora Core (původně RedHat), s důrazem na multimédia, grafiku, lokalizaci a výuku. Tato distribuce je lokalizována do většiny evropských jazyků včetně češtiny. První verze Auroxu (verze 8.0) byla vytvořena v roce 2002 a fungovala na bázi Red Hat 8.0. Aurox používá balíčkovací systém RPM a zachovává kompatibilitu RPM balíčků s Fedorou (dané verze).

WEB Auroxu Získat Aurox

Historie Unixu

Historie unixu začíná v šedesátých letech minulého století v Bellových laboratořích (patřící pod společnost AT&T). V té době probíhá vývoj komplexního operačního systému Multics. Ale protože systém nesplňoval všechny požadavky, měl špatný výkon a také by byl velmi drahý, byl jeho vývoj zastaven. Ken Thompson, který byl z projektu odsunut, pracoval na hře *Space Travel*. Právě při jejím vývoji se začaly vytvářet základy unixu. Počítač, na němž pracovali, byl PDP-7 a měl velmi omezené systémové prostředky. Ona šetrnost je vidět dodnes v názvech jako `cp`, `mv`, `etc`, `usr`. V roce 1970 se ukázalo, že dosavadní PDP-7 je zastaralý, proto Ken Thomspon požádal o nákup nového PDP-11. Bellovy laboratoře projekt podpořily, protože zastavením vývoje Multics neexistoval žádný systém, který by mohly používat.

Jelikož jsou příkazy jednoúčelové, je většinou nutné jich zkombinovat více. Řešením byla roura (pipe, v shellu se zadává takto |), která elegantně přesměrovala výstup z prvního programu na vstup druhého. Tím byla vytvořena základní idea unixu: **Program dělá pouze jednu věc, zato ji dělá perfektně. K dosažení cíle je nutné programy kombinovat. Univerzálním komunikačním prostředkem je text, který se čte ze standardního vstupu a posílá na standardní výstup**.

Původní verze byly napsány v assembleru, ale Ken Thompson toužil po vyšším programovacím jazyce. Jelikož žádný vhodný neexistoval, vytvořil Denis Ritchie programovací jazyk C, do něhož byl kód následně přepsán. Vzhledem k tomu, že napsat kompilátor jazyka C je relativně snadné, bylo možné unix portovat na jiné počítače, což ostatní, v assembleru napsané systémy neuměly.

V polovině sedmdesátých let se unix rozšířil v akademickém prostředí a na univerzitě v Berkeley vznikla BSD (Berkley Software Distribution) a spousta užitečného softwaru. Editor `vi`, tisk, podpora sítí, podpora pro více uživatelů, práce s více procesy,... Postupně došlo k rozštěpení na dvě hlavní větve, BSD a SystemV od AT&T. BSD byl časem uvolněn k volnému použití pod bsd licencí a dnes je z něj svobodný software a je základem systémů jako FreeBSD, OpenBSD, NetBSD, DragonFlyBSD, atd. - některé distribuce GNU/Linuxu jsou dost podobné *bsd např. Gentoo a některé používají bsd-like init např. Slackware.

V roce 1980 vznikla verze Xenix pro procesor intel 8086, za kterou stály firmy Microsoft a SCO (Santa Cruz Operation).

Mezi další známé unixové systémy patří: Solaris od Sun Microsystems, HP-UX od Hewlett-Packard, AIX od IBM.

Časovou osu vývoje různých unixových systémů naleznete na www.levenez.com/unix/history.html (obsahuje spoustu obrázků).

Svobodný software, GNU

GNU

Tento projekt byl započat v roce 1984. Jeho zakladatelem byl Richard Stallman. Komponenty tohoto projektu jsou šířeny pod licencí GPL a .LGPL Cílem tohoto projektu bylo vytvořit volně šiřitelný operační systém postavený na Unixové filozofii.

Nyní je projekt hotov až na jádro Hurd, které je postavené na mikrojádru Mach. Jelikož původní jádro nebylo zatím dokončeno, používá se nyní systém GNU spolu s jádrem Linux, které začal Linus Torvalds. Mimo odbornou veřejnost se toto spojení chybně nazývá pouze Linux, celý název tohoto operačního systému však zní GNU/Linux.

Co znamená GNU?

GNU je rekurzivní akronym pro větu "GNU is not Unix" (GNU není Unix). Gnu však také znamená *pakůň hřivnatý*. Ostatně logo to dokazuje skvěle.

Logo

Autorem loga GNU je Etienne Suvasa. Logo GNU je k dispozici v několika verzích, typech souborů a velikostech.

Free as ...

Říká se: "Free as free speech, not as free beer". Tedy, když mluvíme o svobodném softwaru, mluvíme o svobodě jakožto právu na zdrojový kód, jeho modifikaci a redistribuci. Ale není nelegální si účtovat poplatek za distribuci svobodného softwaru.

Velké množství distribucí GNU/Linuxu je dnes prováděno firmami, ale distribuce jako např. Debian a Ubuntu jsou a budou svobodné, udržované pouze dobrovolníky. Debian je také v ohledu "svobodnosti" výjimečný, má totiž tzv. společenskou smlouvu, v níž se vývojáři zavazují, že distribuce bude obsahovat jen svobodný software a dále stanovují, co je a není svobodné.

Linux

Na počátku devadesátých let minulého století používal jistý Linus Torvalds operační systém Minix. Nespokojen s jeho výkonem a možnostmi začal v roce 1991 psát svoje vlastní jádro. Jeho vývoj oznámil na usenetu v konferenci comp.os.minix. Projekt byl od počátku koncipován jako svobodný, kdokoliv se mohl zapojit a přispět. Záhy se objevilo jméno Linux (Linusův Unix), kterémuž označení se Linus dlouho bránil, ale nakonec je přijal.

V té stejné době projekt GNU obsahoval všechny potřebné nástroje pro operační systém (systémové i uživatelské programy) a proto bylo nově vznikající jádro přizpůsobeno pro GNU a vznikl kompletní operační systém GNU/Linux. Tento pojem se většinou zkracuje na Linux (viz Co je to Linux). První verze byly úzce spjaté s architekturou x86, ale časem došlo k portaci na spoustu jiných architektur.

Katedrála a bazar

Velký rozdíl mezi Linuxem a ostatními jádry ilustruje známá esej Ericha S. Raymonda The Cathedral and the Bazaar (Katedrála a tržiště - česky). Do té doby byla práce na většině OSS projektech v rukou malého množství lidí, kteří programy pečlivě budovali a jednou za čas vydali novou verzi (Katedrála). Linus tohle postavil na hlavu, protože se zapojit mohl každý a nové verze se objevovaly velmi často (Tržiště), takže byl vidět neustálý pokrok. Do té doby nikdo nevěřil, že je možné vytvářet tak komplikovaný kód, jakým je jádro, takovým způsobem.

Složení OS

Operační systém se skládá z několika částí. Nejhlouběji je *jádro*, to se stará o spolupráci s *hardware*m (paměť, procesor, síťové a zvukové karty, pevné disky, apod.) a poskytuje různé služby *procesům*.

Jak vidíte na obrázku, většina programů volá různé funkce knihoven, které se potom předávají jádru. Nicméně některé programy potřebují přistupovat k jádru přímo pomocí jeho systémových volání. A existuje i skupina programů, která vyžaduje přímý přístup k hardwaru (typicky X server). Takové programy musí běžet s právy roota.

Jádro

Jádro je srdcem operačního systému. Zajišťuje komunikaci s hardwarem a poskytuje aplikacím své služby, jako správu procesů, paměti, souborových systémů, podporu sítí atd. Ke komunikaci s hardwarem slouží ovladače. V původních verzích Linuxu bylo jádro monolitické, to znamená, že pro získání podpory jiného zařízení se jádro muselo překompilovat. Dnešní jádro je modulární, potřebné ovladače se ve formě modulů mohou do jádra nahrát, případně z něj zase odstranit.

Knihovny

Anglicky libraries, sg. library, jsou klíčovou součástí systému. Díky tomu, že je GNU/Linux open source, velké množství kódu je právě v knihovnách. Jedná se především o věci standardní, jako jsou např. operace se soubory .jpeg, či matematické funkce jako sin, ale existují například knihovny GTK a Qt, sloužící pro vykreslování tlačítek a dalšího rozhraní. Nikdy tedy neopomíjejte knihovny, jinak se můžete dostat do potíží.

Program, proces

Pojmy *program* a *proces* se často zaměňují.

Program je soubor na disku, který si můžete spustit a on bude vykonávat nějakou činnost. Když ho spustíte, nahraje se do paměti a tím se z něj stane *proces*. Jeden program (např. textový editor `vi`) tedy může být spuštěn více uživateli. Každý z nich má ale svůj vlastní proces.

Zkuste si vypsat seznam procesů ve Vašem sezení:
`ps`

Toto však vypíše pouze procesy v aktuálním shellu. Pro vypsání všech procesů v paměti napište `ps -e`.

Souborový systém

Jako ve většině operačních systémů vycházejících z Unixu, je i v Linuxu jen jeden kořenový adresář, označený '/'. Všechny další souborové systémy se připojují do jednoho stromu, takže když chceme například přistupovat k souborům na disketě, tak ji připojíme do adresáře /mnt/floppy a

pokud máme oddělený diskový oddíl pro adresáře uživatelů, tak jej připojíme do adresáře /home.

Souborové systémy se připojují příkazem `mount` a odpojují příkazem `umount`, nikoliv unmount. Abychom nemuseli ručně připojovat při každém spuštění systému všechny disky a u často připojovaných a odpojovaných souborových systémů, je v `/etc/fstab` uvedeno co a jak se kam připojuje. Ale většina distribucí nastaví při instalaci vše správně a Vy se tak nemusíte o nic starat - jádro je upraveno tak, aby automaticky připojovalo veškerá zařízení.

Struktura souboru /etc/fstab je popsána v jeho manuálové stránce (`man fstab`). Příkaz mount a parametry specifické pro jednotlivé souborové systémy jsou popsány v manuálové stránce příkazu mount.

- Súborové systémy
- Moderní souborové systémy
- Automounter
- Na co se často ptáme: /etc/fstab
- Slovník::/etc/fstab
- FAQ::Souborové systémy

Hierarchie

Strukturu Linuxového filesystemu, který je velice podobný tradičnímu Unixovému, specifikuje tzv. FHS, Filesystem Hierarchy Standard. Tento standard specifikuje, v jakém adresáři mají být jaké aplikace, knihovny či jiné soubory.

S hierarchií adresářů v linuxu nás ve stručnosti seznámí manuálová stránka hier(7).

Kontrola integrity filesystému

O kontrolu souborového systému se nemusíte starat, spouští se automaticky při bootu. Programem `tune2fs` můžete lehce ovlivnit v jakých periodách bude ke kontrole docházet. Rozhodně se nedoporučuje spouštět samotný `fsck` za běhu systému, přesněji řeceno, provádět s ním kontrolu přimountovaného oddílu.

Předcházejme chybám

Je asi samozřejmé, že počítač vypínáme pomocí příkazu `halt` nebo `shutdown`, či pomocí tlačítka v grafickém rozhraní. V případě, že by došlo k silnějšímu zatuhnutí, před stiskem tlačítka reset je vhodno zkusit `Ctrl + Alt + SysRq + F1`, a pak `Ctrl + Alt + SysRq + S`. Tím dojde k synchonizaci disku, a jste v suchu.

Procesy

Proces vlastně není nic jiného, než program, který je spuštěn a běží. Synonymem pro proces může být slovo úloha (task). Linux je od počátku víceúlohový systém, takže v něm může současně[1] běžet více procesů. Stejně jako soubory, tak i procesy jsou vlastněny určitým uživatelem, podléhají přístupovým právům.

Na uživatelské úrovni jsou soubory identifikovány svým jménem (které odpovídá jménu spustitelného souboru), ale systém je jednoznačně identifikuje číslem. Jak je vidět na části příkazu `top`.

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
13743 root      15   0 98.4m  21m 1660 S 12.3  8.5   1:52.07 X
14057 misak     15   0 61884  45m  19m S  4.3  9.0   0:59.27 konqueror
14047 misak     15   0 80420  36m  22m S  2.3  7.5   1:14.01 amarokapp
```

Prvním procesem, který je spuštěn a má pořadové číslo 1 je `init`. Všechny další procesy jsou jeho potomky a jeho zabití vede k ukončení celého systému. Další proces má přiřazeno číslo 2, 3, ..., až po jisté číslo *n*. Jeho hodnota bývá nejčastěji 32768. Skutečná hodnota na vašem systému je uložena v souboru `/proc/sys/kernel/pid_max`. Pokud systém dosáhne maxima, začne přidělovat volná čísla zase od začátku.

Informace o běžících procesech

Jeden příkaz jsem už uvedl a to je příkaz `top`. Je to interaktivní program a slouží k výpisu procesů, který je seřazen podle určitého klíče. Nejčastěji se řadí podle spotřebovaného procesorového času, ale je možné klíč změnit a řadit třeba podle zabrané paměti. Ještě pokročilejší je program `htop` který dokáže procesům posílat signály a podporuje barevné terminály.

Klasičtějším způsobem zobrazení procesů je program `ps`. Tento program je zvláštní tím, že má SystemV a BSD verzi s jinými parametry příkazové řádky. GNU verze tohoto příkazu, která se používá na Linuxu naštěstí zvládá oba dva způsoby. BSD syntaxe je zajímavá faktem, že se v ní parametry neoznačují pomlčkou, jak je tomu prakticky u všech ostatních programů. Protože bylo FreeBSD mým prvním unixem, používám pouze BSD syntaxi.

Příkaz `ps` bez parametrů pouze vypíše procesy spojené s jedním konkrétním terminálem.

```
$ ps
  PID TTY          TIME CMD
30482 pts/6    00:00:00 bash
  583 pts/6    00:00:00 ps
```

Volba `x` zajistí vypsání všech vašich procesů

```
$ ps x
  PID TTY      STAT   TIME COMMAND
13955 ?        Ss     0:00 /bin/sh /usr/kde/3.5/bin/startkde
13982 ?        Ss     0:00 gpg-agent --daemon
...
32128 ?        Ss     0:01 gvim semestralka.tex
  553 ?        S      0:00 ispell -a -S -m -C -d czech
  590 pts/2    R+     0:00 ps x
```

Pokud vás zajímají procesy, které mají spuštěné i ostatní, použijete volbu `a`, ale takto byste získali pouze procesy připojené k danému terminálu. Proto musíme přidat i volbu `x`, která zajistí výpis procesů všech uživatelů na všech terminálech (na něm vidíte, že `init` má skutečně číslo 1).

```
$ ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        S      0:00 init [3]
    2 ?        SN     0:00 [ksoftirqd/0]
...
32128 ?        Ss     0:01 gvim semestralka.tex
```

```
32525 ?        S      0:03 konsole
32526 pts/7    Ss     0:00 /bin/bash
  623 ?        S      0:00 ispell -a -S -m -C -d czech
  649 pts/6    R+     0:00 ps ax
```

Tento výpis není příliš dobrý, protože nezjistíte, kterému uživateli patří který proces. Parametr `u` zajistí podrobnější výpis.

```
$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   1476   396 ?        S    Jan05   0:00 init [3]
root         2  0.0  0.0      0     0 ?        SN   Jan05   0:00 [ksoftirqd/0]
...
misak    32128  0.0  3.3  19084  8520 ?        Ss   18:34   0:01 gvim
semestralka.tex
misak    32525  0.1  6.5  30460 16796 ?        S    18:57   0:03 konsole
misak    32526  0.0  0.6   3544  1776 pts/7    Ss+  18:57   0:00 /bin/bash
misak      623  0.1  4.1  12088 10736 ?        S    19:16   0:01 ispell -a -S -m
-C -d czech
misak      726  0.0  0.3   2896   964 pts/6    R+   19:28   0:00 ps aux
```

Zabíjení

A na závěr si ukážeme jak procesy zabíjet. Dejme tomu, že chceme ukončit program `gvim` s otevřeným souborem `semestralka.tex`. Pokud si necháme vypsat všechny procesy pomocí `ps aux`, jen těžko budeme hledat náš program. Proto můžeme využít rouru (pipe), skrz kterou pošleme výpis programu `ps` do programu `grep` jež výsledek vypíše na `stdout` (obrazovku).

```
$ ps aux |grep gvim
misak    32128  0.0  3.3  19084  8520 ?        Ss   18:34   0:01 gvim
semestralka.tex
```

A následně můžeme procesu poslat zprávu o ukončení. Jako identifikaci použijeme PID procesu:

```
$ kill 32128
```

Pokud chceme ukončit proces dle jména (vhodné při mnoha procesech jednoho programu) , lze využít program `killall`:

```
$ killall gvim
```

Pokud náš program nelze ukončit žádným z obyčejných způsobů a dokonce neodpovídá ani na příkaz `kill` (je zřejmě zacyklený a k vykonání příkazu se nedostane) je možné využít jiné typy signálů vysílaných procesu programem `kill`. Standardně je zasílán signál 15 neboli TERM, který vyvolává exit. Můžete zkusit využít signálu 9, KILL, který má větší šanci, že nebude blokován.

```
$ kill -9 32128
```

Priorita procesu

Priorita procesu se nastavuje příkazem `nice`, který v překladu znamená "ohleduplnost". Hodnota ohleduplnosti je číselné doporučení pro jádro, jak by měl být proces obsloužen ve vztahu k ostatním procesům. Rozsah hodnot pro příkaz `nice` je -20 až +19. Vysoká hodnota znamená nízkou prioritu (budete ohleduplní vůči ostatním uživatelům) a nízká nebo záporná hodnota znamená vysokou prioritu (nebudete ohleduplní).

Nově vzniklý proces dědí hodnotu ohleduplnosti po rodičích. Uživatel může pouze zvyšovat hodnotu ohleduplnosti svého procesu, ale nesmí jí už poté snižovat. Superuživatel root má neomezenou moc v ovládání ohleduplnosti procesů.

Hodnotu ohleduplnosti může uživatel nastavit již při vzniku procesu a při běhu procesů jí může měnit příkazem `renice`. Příkaz `nice` přijímá jako argument příkazový řádek a příkaz `renice` očekává jako argument PID procesu nebo jméno uživatele. Příkazu `renice` se zadává hodnota ohleduplnosti v absolutním tvaru, u příkazu `nice` se ohleduplnost jako inkrement, který se odečte nebo přičte od aktuální hodnoty ohleduplnosti.

```
$nice -n 8 /bin/velmi_narocna_uloha #zvýší ohleduplnost o 8
$renice -3 7623                     #nastaví ohleduplnost na 3
$sudo renice 2 -u jirka             #nastaví ohleduplnost procesů patřící
Jirkovi na 2
```

---

[1]Na jednoprocesorovém stroji může v jeden čas běžet pouze jediný proces, ale díky jejich rychlému přepínání navenek nic nepoznáme.

Komunikace uživatele se systémem

- Shell (terminály atd.)
- GUI
- Prazvláštní metody (braille terminal...)

Příkazová řádka

V unixových systémech se dodnes vedle grafického uživatelského rozhraní (GUI - Graphical User Interface) používá ke komunikaci s operačním systémem hojně a efektivně také tzv. rozhraní příkazové řádky (CLI - Command Line Interface), a to hlavně pomocí programu zvaného shell. Ten příkazy zadané z klávesnice předává operačnímu systému k vykonání.

Většina linuxových systémů má jako výchozí nainstalován shell `bash` (Bourne Again SHell), existuje však řada dalších: `sh, ksh, tcsh, zsh`.

Podrobněji se o programech shell dozvíte v kapitole Shell. Shell

V shellu můžeme pracovat z grafického rozhraní v terminálu (v KDE ho najdeme v menu jako "Konsole" nebo "Terminál", v Gnome "Gnome-terminál" nebo "Xterm" a podobně). Terminálových programů je opět celá řada s různými funkcemi, jejich společný hlavní úkol však je zpřístupnit k práci shell.

Po spuštění terminálu se zpravidla objeví tzv. shellový prompt ukončený značkou `$`:

```
[uzivatel@linuxbox uzivatel]$
```

Můžeme si zkusit do této řádky něco napsat, třeba nějaká nesmyslná písmena, a potvrdit stiskem klávesy `Enter`:

```
[uzivatel@linuxbox uzivatel]$ asdfjklů
```

Pokud všechno šlo dobře, objeví se hlášení:

```
[uzivatel@linuxbox uzivatel]$ asdfjklů
bash: asdfjklů: command not found
```

to jest, `bash` nám zpátky hlásí, že příkaz nebyl nalezen (protože to byl pochopitelně nesmysl).

Nyní stiskněte klávesu se šipkou nahoru a uvidíte, jak se vám předchozí příkaz vrátí. Právě jste zažili, jak funguje historie příkazů. Stiskněte šipku dolů a dostanete znovu prázdný řádek. Takto se můžete pohybovat vpřed a vzad v příkazech, které jste do příkazové řádky napsali.

Šipky vlevo a vpravo použijete pro pohyb kursoru; můžete tak jednoduše opravit případné chyby v příkazu, aniž byste museli mazat celou řádku.

Podrobnější popis a další možnosti viz kapitola Zadávání příkazů

Nyní si probereme tři základních příkazy, které slouží k navigaci souborovým systémem: `pwd` (tj. *print working directory*), `cd` (*change directory*) a `ls` (*list files and directories*).

Odkazy

Velice pěkný článek s názvem Pohádky z příkazové řádky na toto téma napsala Johanka.

Přihlašování

Po naběhnutí systému do textového režimu se zobrazí žádost o login, totiž přihlášení. Přičemž jako první údaj (`linuxbox login:`) zadáme své uživatelské jméno a potvrdíme klávesou *enter*. Údaj druhý (`password:`) znamená heslo. Musíme ho psát "poslepu", neboť se většinou z důvodů bezpečnosti nezobrazují ani hvězdičky. Oba dva údaje nám dá správce, či jsme si je vyplnili sami při instalaci. Jsou-li správné, proběhne přihlášení. Pokud se nám podaří překlep v jednom či druhém, můžeme pokus opakovat.

Úspěch vypadá následovně: `[uzivatel@linuxbox uzivatel]$` nebo taky: `uzivatel@linuxbox:~$` a podobně. Příkazová řádka je nám k dispozici.

Neexistuje-li uživatelský účet a současně máme přístup superuživatele, přihlásíme se jako *root*. V tomto případě jako další krok založíme neprivilegovaného uživatele na průzkum systému i každodenní práci. FIXME

`/etc/issue` - obsah tohoto souboru se vypíše ještě před vlastním přihlášením, obvykle obsahuje základní informace o systému, např. verzi jádra, název stroje.

`/etc/motd` - obsah tohoto pak po úspěšném přihlášení, doslova znamená "messages of the day" a lze ho použít např. k zobrazování náhodných tipů či hlášek pomocí `fortune`.

Výše zmíněné soubory můžeme upravovat pod superuživatelským účtem. V některých distribucích je nutno ještě zamezit opětovnému samovolnému přepsání při rebootu zásahem do startovacích skriptů.

Terminály

Terminál bylo kdysi místo na okraji sálového počítače, kde seděl pověřený pracovník a komunikoval s počítačem pomocí klávesnice a tiskárny, později monitoru. V dnešní době je terminálem vlastně každý desktopový počítač. Vy, jako uživatel u něj sedíte, a používáte vstupní a výstupní zařízení (klávesnici, monitor, sluchátka).

Virtuální terminály

Ačkoli by se mohlo zdát, že jde o nějaké sci-fi, jako virtuální realita, opak je pravdou. Virtuální terminály jsou na každém GNU/Linuxovém systému. A kolik jich tam je, ptáte se? Přesně 12. Jako by jste měli několik počítačů. Terminály jsou zcela odděleny, do každého může být přihlášen někdo jiný. Přepíná se mezi nimi pomocí `Ctrl + Alt + F1` až `Ctrl + Alt + F12`. Prvních šest je obvykle textových a ty zbývající jsou grafické.

Zadáváni příkazů

Příkazová řádka v Linuxu je opravdu mocný nástroj. Umožňuje nám libovolně manipulovat se systémem. Narozdíl od některých operačních systémů, kde je příkazová řádka jen doplňkovým nástrojem. V případě Linuxu se ovšem jedná o jeden ze základních nástrojů, tomu také odpovídají její možnosti.

Kouzelná klávesa TAB

Při pročítání diskusí, návodů a jiných článků o Linuxu, často narazíte na rady, které očekávají, že budete zadávat dlouhé příkazy. Pokud požíváme BASH, pak nám mnoho námahy ušetří právě klávesa TAB. Pokud ji stiskneme na prázdné příkazové řádce, vypíše se nám seznam všech příkazů. (Ještě předtím jsme však dotázáni, zda-li to nebyl překlep `Display all 1774 possibilities? (y or n)` To však není jeho hlavní použití, mnohem užitečnější je zadat několik prvních písmen příkazu např. budu-li chtít spustit příkaz ifconfig, napíši pouze if a stisknu TAB.

```
holly:/home/geo/net# if
if        ifconfig  ifdown    ifup
```

V systému existuje více příkazů, které začínají písmenky if, TAB tedy vypsal všechny možnosti. Já chtěl spouštět ifconfig, zadám tedy další písmenko, tedy c, a opět stisknu TAB. Příkaz se doplní a já už mohu stisknout enter. Mohu vás ujistit, že po několika dnech po objevení této klávesy si začnete pamatovat většinu příkazů jen podle jejich začátků. Doplňování samozřejmě funguje i pro názvy souborů. Tedy chci například rozbalit nové jádro. Soubor se jmenuje linux-2.6.11.6.tar.bz2. Zadám tedy

```
$ tar -xjvf l
```

a stisknu klávesu TAB. Ve svém adresáři mám však více souborů od l, proto se objeví.

```
lgp-30-man.html
linuxlgp-30-man_soubory/
linux-2.6.11.6.tar.bz2
```

zadám tedy "i" a název se doplní na linux, zadám tedy -. Zadání je teď jednoznačné a bash již doplní na celý název.

Historie

Nemám na mysli dějepisectví, ale historii příkazové řádky. Pomocí kláves `šipka nahoru` a `šipka dolu` se můžete lehce pohybovat ve vašich posledních příkazech. Také lze využít vyhledávání v historii. Například pokud víte, že včera jste napsali velice dlouhý příkaz a nechcete ho vymýšlet znovu a prohledávat historii příkazů pomocí šipky nahoru je moc zdlouhavé, můžete využít hledání v bash historii - CTRL+r. Budete psát libovolnou část příkazu, dokud se neobjeví ten, který jste měli na mysli. Úspěšnost hledání závisí samozřejmě na nastavené velikosti historie.

&&

Tento výraz odpovídá logickému operátoru AND (tedy česky "a"). Používá se k vytvoření řetězu příkazů, které se mají vykonat hned po sobě. Například: `mount /mnt/fd && cp -r /mnt/fd/ /home/uzivatel/neco/ && umount /mnt/fd` . Je to podobné, jako kdybyste zadali výše jmenované příkazy ručně po sobě. Příkazy je také možné oddělovat jen středníkem (tedy např.: mount /mnt/fd; cp -r /mnt/fd/ /home/uzivatel/neco/; umount /mnt/fd), ale operátor && zajistí, že následný příkaz v řetězci bude vykonán jen a pouze tehdy, pokud ten předchozí neskončí chybou (v případě středníku nezáleží na tom, zda předchozí příkaz skončil s chybou).

||

Tento výraz odpovídá logickému operátoru OR (česky "nebo"). Lze jím stejně jako operátorem && spojovat několik příkazů za sebou, přičemž ovšem následující příkaz bude proveden pouze tehdy, pokud předchozí skončí chybou (tedy přesný opak operátoru &&).

GNU Readline

Jedná se o backend (to co nevidíme, ale je to nutné). Knihovna readline implementuje funkce, které nám například umožňují psát příkazy a procházet historií v BASHi

Standardní vstup a výstup

Standardní vstup (neboli **stdin**) je místo, ze kterého programy berou data a standardní výstup (**stdout**) je místo, kam je vypisují. Příkaz `cat` bez parametrů nedělá nic jiného, než že čte data ze standardního vstupu a vypisuje je na standardní výstup. Zkuste v konzoli napsat `cat` a stisknout enter. Poté napište jakoukoliv větu a po stisku enteru se vám zobrazí na obrazovce. Ukončíte stiskem Ctrl+D.

```
$cat
Standardní vstup je připojen na klávesnici
Standardní vstup je připojen na klávesnici
Standardní výstup je připojen na monitor
Standardní výstup je připojen na monitor
```

Na výpisu programu vidíte, že standardním vstupem je vaše klávesnice a standardním výstupem je obrazovka monitoru. Ale ne vždy tomu tak musí být (viz další část přesměrování a roury). Programy, které se takto chovají, nazýváme filtry, protože nejčastěji slouží k úpravám a filtracím textů.

Standardní chybový výstup

Mimo tyto dva existuje ještě chybový výstup (**stderr**), do něhož jsou vypisovány chybová hlášení. I on je standardně vypisován na monitor. Smysl jeho existence je ve snadném oddělení užitečného výstupu programu od chybových hlášení, či varování.

Přesměrování a roury

Ne vždy musí být standartním vstupem klávesnice a výstupem obrazovka – to díky přesměrování. Předpokládejme, že máme soubor `foo` a v něm nějaký text. Příkaz `cat` nečte jen ze standardního vstupu, ale dokáže přečíst vstupní soubor(y) a ty pak zobrazit na standardní výstup.

```
$ cat foo
stdin
stdout
stderr
$ cat foo > bar
```

Počkat, když jsme napsali `> bar`, tak se nic nezobrazilo! A to proto, že znak > říká shellu, aby standardní výstup nevytiskl na monitor, ale zapsal do souboru. Jinými slovy jej **přesměroval**. Když napíšete `cat bar`, zobrazí se stejný obsah, jako je v souboru foo.

Znak > (respektive `1>`) přesměruje standardní výstup do souboru. Znak `2>` přesměruje standardní chybový výstup do souboru a `&>` přesměruje oba dva proudy do stejného souboru. Podobně < umožňuje přesměrovat obsah souboru na standardní vstup. Dvojité **>>** pak místo přepsání souboru přídává data na jeho konec.

Roury

Zatímco teď jsme přesměrovávali výstup do souboru, roury přesměrovávají výstup na standardní vstup jiného programu. Roury způsobují, že je používání shellu tak mocné, protože umožňují kombinovat více filtrů do jedné kolony. Dejme tomu, že máme seznam jmen v souboru lide.txt, který chceme setřídit podle abecedy, vyřadit duplicity a zobrazit prvních 10 lidí.

$ sort -u lide.txt | head

Program `sort` setřídí seznam lidí a s parametrem `-u` také data zbaví duplicit. `Sort` předá data dál a příkaz `head` vytiskne prvních deset řádků. Někteří lidé to přirovnávají k lidské řeči. Máme spoustu slov (příkazů), které samy o sobě označují (vykonávají) pouze jednu činnost. Ale díky tomu, že je můžeme kombinovat do vět (kolon), můžeme pomocí těch jednoduchých slov (příkazů) vyjádřit složitější myšlenky (provádět složitější činnost).

Hrátky s přesměrováním

Možná jste už slyšeli o adresáři `/dev`. Pokud ne, tak vězte, že v tomto adresáři se schovávají různá reálná (`/dev/hda`) i nereálná (`/dev/random`) zařízení (přesněji, jejich souborové reprezentace).

Ty, které se používají k přesměrování si popíšeme:

- **/dev/null** – je to taková malá **černá díra**, cokoliv tam pošleme, tak se ztratí. Používá se například k filtrování hlášení, která nás nezajímají (`cat zadny_soubor 2> /dev/null` - pokud soubor neexistuje, cat vypíše chybu, ale tím že jsme ji přesměrovali do /dev/null ji neuvidíme)
- `/dev/stdin` - reprezentuje **standardní vstup**. Používáme v případě, když program nečte ze standardního vstupu, ale pouze ze souboru, příkladem je `echo "Žluťoučký kůň" | iconv -f iso-8859-2 -t utf-8 /dev/stdin`. Tím můžeme programy používat v koloně i v případě, že s tím autoři nepočítali.
- **/dev/stdout** - **standardní výstup**, pokud chcete v koloně zpracovat i chybový výstup, napíšete `cat zadny_soubor 2> /dev/stdout`
- `/dev/stderr` - **standardní chybový výstup**, příkaz echo posílá vstup na stdout, ale můžeme to přesměrovat `echo "chyba" > /dev/stderr`, což je správný způsob vypisování chybových hlášení.

Shell

Shell je textové rozhraní pro komunikaci mezi uživatelem a systémem. Shellů je několik desítek, mezi nejznámější patří bash, dash, zsh a csh. GNU/Linuxové distribuce obvykle upřednostňují bash, zatímco *BSD spíše csh, nebo jeho rozšířenou a zpětně kompatibilní "nadstavbu" - tcsh. Pokud chcete zvolit jiný shell než jaký právě používáte, jako root použijte `chsh`, ale pozor, pokud se překlepnete, a nastavíte si shell pro superuživatele nebo jiného uživatele na neplatnou cestu, dotyčný se nebude moci přihlásit! :(

Bash

`uzivatel@localhost:~$` - takto, nebo velmi podobě Vás bash přivítá. Místo `uzivatel` bude vaše uživatelské jméno a localhost bude název počítače. (V případě, že před tím vyplňujete username a password, tak to ještě nejste v shellu.) Práce s bashem je velice jednoduchá - zadáte příkaz a odentrujete. Pro orientaci v příkazech poslouží Přehled příkazů a sekce Principy práce se systémem. Pokud změníte složku mimo svou domovskou, bash vypíše `uzivatel@localhost:/slozka$` V případě, že jste přihlášen jako root (superuživatel), bash se bude hlásit `localhost:/slozka/kde/jste#`. Způsob, jak se vám bash hlásí se dá změnit, takže se vám pak systém může hlásit například smajlíkem - toto nazýváme prompt.

V případě, že zadáte příkaz jehož vykonání si žádá čas, tak vám kurzor poskočí na nový řádek a tam zůstane a až po dokončení onoho příkazu se znovu vypíše prompt - v žádném případě tedy není důvod k panice ve stylu "já to zkazil".

Tcsh

Tcsh vznikl rozšířením csh z bsd unixu v roce 2001. Je s csh zpětně kompatibilní a pokud nemáte v `~/.tcshrc` soubor, bude používat nastavení ze souboru `~/.cshrc`. Syntaxe tcsh/csh je trochu odlišná od toho co znáte v bashi:
tcsh - `alias la ls -a`
bash - `alias la='ls -A'`

Dokumentace

V běžné distribuci Linuxu naleznete stovky, nebo tisíce programů, které můžete použít. Nikdo není schopen si pamatovat, co který dělá a jaké má parametry. Proto existuje dokumentace, do níž může člověk nahlédnout.

Manuálové stránky

Jsou standardní dokumentací. Pravděpodobně každý zkušenější uživatel se nejprve podívá sem. Ke čtení dokumentace slouží příkaz `man`. Manuálové stránky mají ustálenou strukturu, například upravený výpis `man cp`.

```
CP(1)                                                                CP(1)

NAME
       cp - copy files and directories

SYNOPSIS
       cp [options] file path
       cp [options] file... directory
DESCRIPTION
       cp copies files (or, optionally, directories).  You can either copy one
       file to a given destination, or copy arbitrarily many files to a desti-
       nation directory.
ENVIRONMENT
       The  variables  LANG, LC_ALL, LC_COLLATE, LC_CTYPE and LC_MESSAGES have
       the  usual  meaning.  For  the  GNU   version,   the   variables   SIM-
       PLE_BACKUP_SUFFIX  and  VERSION_CONTROL  control backup file naming, as
       described above.
SEE ALSO
       mv(1)
```

První řádek označuje jméno příkazu (`cp`) a číslo sekce manuálových stránek. V části `NAME (JMÉNO)` se dozvíte stručný popis činnosti programu. V tomto případě, že příkaz cp kopíruje soubory a adresáře. Zajímavější je část `SYNOPSYS (SYNTAXE)`, která označuje povinné a nepovinné parametry příkazu. Všechno v hranatých závorkách je nepovinné. Je nutné zadat buďto soubor a cestu, kam se zkopíruje, nebo více souborů a nakonec adresář, kam se soubory zkopírují. Část `DESCRIPTION (POPIS)` obsahuje podrobnější slovní popis funkce programu. V části `ENVIROMENT (PROSTŘEDÍ)` se dozvíte, které systémové proměnné ovlivňují běh tohoto programu. Užitečná je funkce `SEE ALSO (VIZ TÉŽ)`, v níž se dozvíte o ostatních podobných programech.

Pokud chcete zobrazit informace k danému tématu, napište `man -k jmeno_tematu` , třeba informace o PostScriptu zobrazíte pomocí `man -k ps` .

GNU info stránky

Formát manuálových stránek je nevhodný pro rozsáhlou dokumentaci, která je pro některé programy nutná. Lidé z GNU proto zavedli formát info, které se prohlíží stejnojmenným programem. Info stránky jsou rozsáhlejší, strukturované (info stránky podporují hypertextové odkazy). Spousta manuálových stránek obsahuje upozorněním, že více toho je na info stránce programu.

Různé prohlížeče

Existuje více prohlížečů manuálových stránek, než klasický `man`. Program Konqueror, který je součástí prostředí KDE, umožňuje prohlížení manuálových stránek v grafickém prostředí. Zadejte do adresního řádku `#příkaz`, nebo `man:/příkaz` a můžete prohlížet dokumentaci stejně komfortně, jako na webu. V sekci `SEE ALSO` dokonce najdete hypertextové odkazy na další manuálové stránky. Úplně stejně funguje i čtení GNU info stránek, pouze prefix je `info:/příkaz`.

Principy práce se systémem

Soubory v Linuxu

Soubor je v podstatě určitá posloupnost bajtů, která k sobě jistým způsobem patří. Příkladem souboru je html stránka, spustitelný program, obrázek, film, ... Prakticky každý program vytváří, nebo čte nějaké soubory (webový prohlížeč, přehrávač hudby, textový editor, ...). Spousta z nich čte soubory, které odpovídají určité struktuře, ale základní unixové utilitky se na soubory dívají jako na proud bajtů.

Pokud se chceme podívat na obsah nějakého souboru, není nic jednoduššího, než napsat

```
cat linux.txt
Mandriva
Aurox
Kubuntu
Slackware
```

Když se podíváte do manuálové stránky, zjistíte, že `cat - concatenate files and print on the standard output`. Cat tedy neznamená kočka, ale je to zkratka ze slova con**cat**enate - zřetězit, spojit. Tento příkaz tedy spojuje soubory.

```
cat linux.txt bsd.txt
Mandriva
Aurox
Kubuntu
Slackware
FreeBSD
OpenBSD
NetBSD
```

Stránkovací programy

Popravdě, málokteré soubory, s nimiž se setkáte budou tak krátké, aby se vešly do konzole. Proto byly vytvořeny stránkovací programy, které výstup po stránce zastaví a umožní vám si vše v klidu přečíst. Nejstarším příkazem je `more`, který je velmi jednoduchý. Prostě zastaví stránku a stiskem mezerníku se dostanete na další stranu a stiskem Enteru na další řádek. Zadáním znaku '/' (jenom lomeno, bez uvozovek) se dostanete do režimu hledání (stejné je to i ve vi). Klávesou 'q' ukončíte jeho činnost. Program `less` vznikl jako dokonalejší náhrada programu `more`. Zatímco prvně jmenovaný umožňoval pouze pohyb dopředu, `less` umožňuje i pohyb zpět. Ovládá se stejným způsobem, ale pohyb mezi stránkami může být ovládán klávesami Page up a Page Down. Ještě dokonalejší náhradu představuje program `most`.

Tyto programy se dají použít třemi způsoby. Buďto přímým zadáním `more foo.txt`, nebo za rouru `cat foo.txt | most`. Posledním případem je nastavení systémové proměnné `PAGER`, která nastavuje výchozí stránkovací program, který je použit, pokud je potřeba zalomit dlouhý výstup.

```
echo $PAGER
/usr/bin/less
```

Určení typu souboru

Nejjednodušším způsobem, jak můžete určit typ souboru je příkaz `file`. Ten používá takzvané *magic files* (jsou v `/usr/share/misc/file/`), což jsou soubory obsahující charakteristické znaky jednotlivých souborů (zalomeno).

```
file /etc/passwd /bin /usr/bin/file /dev/cdrom /dev/hdd /tmp/
/etc/passwd:   ASCII text
/bin:          directory
/usr/bin/file: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for
GNU/Linux 2.4.1, dynamically linked (uses shared libs), stripped
/dev/cdrom:    symbolic link to `hdd'
/dev/hdd:      block special (22/64)
/tmp/:         sticky directory
```

Příkaz `file` je velmi mocný. S jeho pomocí například velmi snadno zjistíte, jakým kodekem je komprimován film.

```
file *avi
foo1.avi: RIFF (little-endian) data, AVI, 320 x 240, 23.98 fps, video:
DivX 3 Fast-Motion, audio: MPEG-1 Layer 3 (stereo, 44100 Hz)
foo2.avi: RIFF (little-endian) data, AVI, 576 x 432, 25.00 fps, video:
XviD, audio: MPEG-1 Layer 3 (stereo, 48000 Hz)
foo3.avi:   RIFF (little-endian) data, AVI, 320 x 240, 25.00 fps, video:
DivX 4, audio: MPEG-1 Layer 3 (stereo, 44100 Hz)
```

Speciální soubory

Ne všechny soubory v Linuxu jsou pouze dokumenty, nebo aplikace. Unixové chápání souboru je daleko širší, než jste dosud byli zvyklí. To, s jakým souborem máte čest zjistíte příkazem `ls -l` (více o tomto příkazu zjistíte z následující kapitole Přístupová práva).

```
ls -l
total 36
brw-rw----  1 root root 13, 0 Mar 10  2005 block-device
crw-rw----  1 root root 83, 0 Mar 10  2005 character-device
drwxr-xr-x  2 root root  4096 Oct  2 17:39 directory
-rw-r--r--  2 root root 14830 Oct  2 17:39 hardlink
prw-r--r--  1 root root     0 Oct  2 17:39 named-pipe
-rw-r--r--  2 root root 14830 Oct  2 17:39 regular-file
srw-rw-rw-  1 root root     0 Oct  2 15:35 socket
lrwxrwxrwx  1 root root    12 Oct  2 17:38 symlink -> regular-file
```

Jak vidíte, existuje mnoho druhů souborů

- **Normální soubor** (regular file), je označen znakem **-** a je to klasický soubor, na který jsme zvyklí (dokument, aplikace, obrázek, nebo cokoli jiného).
- **Adresář** (directory), označuje znak **d**. V unixovém pojetí je adresář speciálním druhem souboru.
- **Symbolický odkaz** (symbolic link, symlink), označujeme znakem **l** a dá se přirovnat k hypertextovému odkazu na webu. Ve výpisu vidíme, na který soubor odkaz ukazuje. Odkaz je svázán se jménem souboru, tudíž jeho přejmenování odkaz zneplatní. Můžeme vytvářet odkazy na adresáře. Vytváříme příkazem `ln -s`.
- **Pevný odkaz** (hardlink), není nijak označen. Je to prostě jiné jméno pro soubor (který je na disku pouze jednou). Ve výpisu vidíme, že soubory `regular-file` i `hard-link` mají ve druhém sloupci dvojku. Což je počet pevných odkazů. Pevné odkazy nejsou svázány se jménem souboru (ale s číslem inodu, pokud to musíte vědět). Nejde dělat pevné odkazy mezi různými oddíly disku a na adresáře. Vytváříme příkazem `ln`.
- **Blokové a znakové zařízení** (Block and character device) je označeno znakem **b** (respektive **c**). Vyskytují se v adresáři `/dev` a jsou to souborové reprezentace blokových (pevný disk) a znakových (terminál) zařízení, která jsou připojena k počítači. Vytváříme ručně příkazem `mknod`, nebo se o ně stará `udev`, či starší `devfs`.
- **Pojmenovaná roura** (named pipe) - o tom, co to vlastně roura je, se dozvíte v kapitole (Přesměrování a roury). Pomocí roury mohou komunikovat programy mezi sebou. Výhodou je, že se tváří jako soubory (přestože se nejčastěji jedná o místo v paměti). Vytváříme příkazem `mkfifo`.
- **Socket** (Socket) - slouží ke komunikaci mezi procesy a to buďto na lokálním stroji, nebo mezi vzdálenými stroji, po síti. Narozdíl od rour podporují oboustrannou komunikaci.

Přístupová práva

Linux je víceuživatelský systém a je proto nutné zajistit, aby mi ostatní nemohli měnit, číst moje dokumenty. Pokud jim to tedy výslovně nepovolím. Každý soubor (i adresář) v Linuxu má přidělená práva, která upravují práci se souborem. Příkaz `ls` vypisuje obsah zadaného adresáře. Přídáním parametru `-l` si zobrazíme podrobnější informace, včetně práv.

```
ls
bsd.txt  Desktop  dnsping  linux.txt
ls -l
celkem 16
-rw-r--r--  1 misak users   23 zář  1 23:06 bsd.txt
drwxr-xr-x  2 misak users 4096 zář  1 23:08 Desktop
-rwxr-xr-x  1 misak users  443 zář  1 23:08 dnsping
-rw-r--r--  1 misak users   33 zář  1 23:05 linux.txt
```

Práva jsou uvedená v prvním sloupci. První znak označuje typ souboru, pomlčka '**-**' označuje normální soubor, '**d**' adresář, dále '**l**' je symbolický odkaz, '**c**' znakové zařízení, '**b**' blokové zařízení a '**p**' je pojmenovaná roura (pokud nevíte, co jednotlivé typy znamenají, nic se neděje, prostě čtěte dále).

Za typem je potom devět znaků, které symbolizují práva. Vždy jsou vypsány v pořadí právo pro čtení '**r**', právo pro zápis '**w**' a právo pro spuštění '**x**'. Na úrovni adresářů '**r**' značí, že je možné vypsat seznam souborů, '**w**', že je možné do adresáře soubory přidávat nebo je z něj odebírat a '**x**', že adresář smí být součástí cesty. První trojice je nastavení platné pro vlastníka souboru (třetí sloupeček - misak), druhá pro skupinu (čtvrtý - users) a poslední pro ostatní. Vlastník souboru je uživatel, který jej vytvořil (nebo bylo na něj vlastnictví převedeno příkaze `chown`) a obvykle má největší pravomoce. Skupina usnadňuje sdílení souborů mezi uživateli. Můžeme zvýšit práva lidem jen ze svojí skupiny a ne těm, kteří tam nepatří.

Spustitelné soubory

Linux bere soubor jako spustitelný jen tehdy, když má nastavená práva pro spuštění. Tento systém má své kouzlo v tom, že spustitelný soubor může být jakéhokoli typu (binární soubor, skript v shellu), ale z hlediska jeho spouštění je to úplně jedno. U skriptů je podmínka, aby měly na prvním řádku uvedený interpret, který je bude provádět (`#!/bin/sh`, nebo přenositelněji `/usr/bin/env python`) a binární soubor musí být podporován jádrem (nejpoužívanější je elf, starším formátem je a.out).

Další práva

Soubor, který je uložený v adresáři, do něhož mají všichni povolen zápis může také kdokoliv smazat. Někdy se hodí, kdyby jej mohl mazat pouze vlastník souboru a nikdo jiný. Typickým představitelem je adresář `/tmp`. Tento adresář má nastavený **sticky** (lepkavý) bit, který se nastavuje příkazem `chmod +t` a ve výpisu příkazu `ls` je vidět takto:

```
drwxrwxrwt   39 root  root   3704 zář  6 21:15 tmp
```

Jěště existují takzvané *set* bity. Pokud má soubor nastaven setuid (resp. setgid bit), pak při spuštění programu dostane dočasně program práva uživatele (resp. skupiny), jakou má vlastník souboru. Normáně by program pracoval pod právy uživatele, který jej spustil. Proto se tato technika používá u dočasného zvýšení práv. Například pokud je vlastník souboru /bin/ping root, pak ať jej spustí kdokoli, program běží pod právy superuživatele (protože potřebuje přistupovat k prostředkům, které nejsou obecně dostupné obyčejným uživatelům. Setuid se nastavuje pomocí `chmod u+s`, setgid pak díky `chmod g+s`.

Program spuštěný uživatelem má stejná práva, jako uživatel sám. To znamená, může modifikovat jen ty soubory k nimž má uživatel právo zápisu, ... Existuje několik případů, kdy je nutné, aby uživatelem spuštěný program běžel pod jiným účtem (zpravidla s vyššími pravomocemi). Typickým příkladem je příkaz `passwd` pro změnu hesla uživatele. Ten musí modifikovat soubor `/etc/shadow`, kde jsou uložené hashe hesel. Tento soubor není, z pochopitelných důvodů, zapisovatelný pro běžné uživatele. Program passwd má nastaven příznak `suid` a vlastní jej uživatel `root`. Program po svém spuštění má práva uživatele root, přestože jej spustil jiný uživatel. Příznak se nastaví příkazem `chmod u+s`.

```
ls -l /bin/passwd
-rws--x--x  1 root root 32108 čec 11 14:30 /bin/passwd
```

Existuje také podobný příznak setgid, který mění skupinu. Není tolik používán u souborů, ale u adresářů, kdy po aplikaci tohoto příznaku patří všechny nově vytvořené soubory do určené skupiny a ne do uživatelově hlavní. Nastaví se příkazem `chmod g+s` a výpis vypadá následovně.

```
drwxr-sr-x   3 root users     4096 zář  1 23:14 foo
```

Číselné vyjádření

Práva jsou na systémové úrovni reprezentována jako čtyři čísla v rozsahu od 0 až po 7 (tj. tři bity). Tyto číslice reprezentují speciální práva, uživatelská, skupinová a ostatních. Právo pro čtení má hodnotu **4**, pro zápis **2** a pro spuštění **1**. Jejich součtem potom získáme součtem těchto hodnot. Hodnota **7** (4+2+1) označuje všechna práva, hodnota **5** (4+1) značí právo pro čtení a spuštění. Hodnota **0** pak, celkem logicky, určuje žádná práva.

Speciální práva jsou ohodnocena následovně. **4** je setuid, **2** je setgid a **1** je sticky bit.

Kompletní nastavení adresáře `/tmp` se tedy dá provést

```
chmod a+rwx /tmp        # zdlouhavejsi zpusob
chmod +t /tmp
chmod 1777 /tmp         # rychlejsi zpusob
```

Změnu vlastníků lze pak provádět pod rootem následně:

```
chown -cR uzivatel:skupina /tmp      # -cR vztáhne změnu i na adresáře
```

ACL

Jemnější nastavení práv umožňuje ACL, viz příkazy `getfacl` a `setfacl`.

Jména souborů

V Linuxu a ostatních UNIXových operačních systémech se u názvů souborů a adresářů rozlišuje velikost písmenek. `foo, FOO, Foo` jsou v Linuxu tři zcela rozdílné soubory. Název může obsahovat jakékoliv znaky s výjimkou lomítka /, ale nejlepší je používat **písmena, čísla, pomlčku, podtržítko a tečku**. Pokud chcete použít v názvu souboru mezeru, je třeba uzavřít název souboru do uvozovek nebo použít zpětné lomítko před danou mezerou. Zápis pak vypadá `"soubor s mezerami.txt"` případně **soubor\ s\ mezerami.txt**. Každopádně se nedoporučuje používat v názvech mezery a jiné speciální znaky. Také se vyhněte háčkům a čárkám, mohli byste s nimi mít problémy v jiných operačních systémech nebo v některých nelokalizovaných programech. Délka názvu může být několik set znaků, ale takové názvy snižují čitelnost a špatně se s nimi pracuje.

Speciální význam mají soubory, které začínají tečkou. Ty se v běžném výpisu příkazu `ls` nevyskytují a obvykle slouží k uložení konfigurace (např. `.bashrc`). Je kvůli tomu, že se s konfiguračními soubory často nepracuje a v normálním výpisu by pouze překážely. Příkaz `ls` s parametrem `-a` nám zobrazí i tyto skryté soubory:

```
ls -a ~
.                       .kodos
..                      .kpackage
.adobesvg               .licq
.alsaplayer             .links
.bash_history           .mailcap
```

Adresář '**.**' je odkazem na aktuální adresář a '**..**' je odkazem na nadřazený adresář (jedinou vyjímkou je kořenový adresář '/', ten nemá nadřazený adresář, proto je '..' odkazem na '/')

Konfigurace obecně

Všechny kritické a důležité programy pro linux si ukládají svoje konfigurační soubory do `/etc`. Ostatní a méně důležité programy si ukládají své konfigurační soubory přímo do domovského adresáře uživatele, pod kterým je spuštěn (např. /home/uzivatel/.mplayer/config). Mějte prosím na paměti, že každá distribuce může mít strukturu v `/etc` trochu jinou nebo některé soubory jinak pojmenované. Zde si popíšeme některé hlavní soubory, které by nás mohly zajímat.

```
X11                     # nastavení X serveru je v /X11/xorg.conf
acpi                    # konfigurace acpi, včetně definování tlačítek
conf.d                  # zde jsou uložena nastavení dalších programů
cups                    # nastavení tiskového serveru
dbus-1
dhclient.conf
dhcpd.conf
fam
fb.modes                # nastavení framebufferu
fdprm
fonts                   # konfigurace fontů
fstab                   # konfigurace disků a souborových systémů
group                   # konfigurace skupin
group.lock
gshadow
gshadow-
gshadow.lock
gtk
gtk-2.0
hal
hibernate               # nastavení hibernace počítače
hosts                   # uloženy IP adresy a jména počítače
hotplug                 # konfigurace Plug and Play
inittab
iptables                # nastaveni firewallu (založeného na IPtables)
ld.so.conf              # hlavní knihovny
lilo.conf               # konfigurace zavaděče
localtime               # časové pásmo
mime.types              # definované přípony souborů a akce s nimi
modprobe.conf           # moduly, které se budou nahrávat při startu PC (jádro
                        2.6)
modules.conf            # moduly, které se budou nahrávat při startu PC (jádro
                        2.4)
mtab                    # uložena konfigurace disků (fstab a mtab jsou spolu
                        propojeny)
ntp                     #  synchronizace času přes internet
pam.d                   # autorizace uživatelů
passwd                  # hesla uživatelů (šifrována)
pcmcia                  # konfigurace pcmcia
```

```
printcap                # tiskárna z CUPS
profile                 # profily (zejména cesty programů)
resolv.conf             # nastavení nameserveru
sane.d                  # nastavení skeneru (uživatel má vlastní)
sensors.conf            # konfigurace sensorů a čidel (teplota, otáčky atd.) z
lm_sensors

sysconfig               # některé distribuce si do tohoto adresáře ukládají
                        nastavení lokalizace,místních zvyklostí a mnoho dalších.
                        Stojí za Vaši pozornost, pokud existuje, protože většina
                        potřebných a užitečných nastavení je právě tam.
ssh
ssl
sysctl.conf
syslog-ng
timezone                # časové pásmo
udev                    # konfigurace připojení periférií uživatelem
(udev+fstab+hal+dbus= Plug and                          Play známe ve
Windows)

xinetd.conf
xinetd.d
```

Máme dvě možnosti konfigurace - přímo ručně ditovat konfigurační soubor, nebo použít některý z nástrojů, který námi zvolená distribuce nabízí.

Rozsáhlé a komplexní sady programů, jako KDE,Gnome a podobné mají své vlastní grafické nástroje. Pokud ale chcete konfigurovat např. Apache, uchýlíte se nejspíše k možnostem vaší distribuce, pokud toto nastavení nabízí. Je většinou velmi srozumitelné i pro začátečníky a v češtině. Nezapomínejme, že vše je soubor. Veškerá konfigurace GNU/Linuxu je tudíž také v souborech, nikoliv ve zvláštních chaotických strukturách jako registry MS Windows.

Pamatujte že každý zásah do /etc může poškodit Váš systém!!

Specifické konfigurace jednotlivých distribucí

Debian
V distribuci Debian nám poslouží `dpkg-reconfigure nazev-baliku`, například `dpkg-reconfigure xserver-xorg`. Pokud má daný balíček něco ke konfiguraci, dostanete možnosti konfigurace, jinak se nenapíše nic. /etc/apt/sources.list - pokud má vaše distribuce balíčkovaní přes apt (Debian, Ubuntu), zde nastavíte servery, nebo lokální cesty, z nichž se bude snažit získavat balíky a jejich seznamy - doporučuji `man sources.list`

SUSE,Novell
Suse a Novell používá komplexní nástroj Yast.
Mandriva Linux
Mandriva Linux používá také komplexní nástroj a to Mandriva Control Center.
Arch Linux
Nejdůležitější nastavení jsou uložena v rc.conf !

Balíčkovací systémy a instalace softwaru

Obecně

Velmi častým dotazem začínajícího uživatele Linuxu je „Jak mám ten program nainstalovat?". Způsob instalace software v Linuxu můžeme rozdělit do dvou základních kategorií.

- instalace z balíčku
- instalace ze zdrojových kódů.

Poslední záchrana: Instalace ze zdrojových kódů

Nejdřív si povíme něco o instalaci ze zdrojových kódů. Většina programů používaných v Linuxu je k dostání i ve formě zdrojových kódů, které si může uživatel sám upravovat a překládat do výsledné spustitelné podoby. Editovat je lze libovolným textovým editorem. Pro překlad je třeba určitch zkušeností a znalostí operačního systému. Obecně se moc začínajícím uživatelům překlad ze zdrojových kódů nedoporučuje, protože velmi rychle zapomínají co a kam nainstalovali, nemají představu které soubory patří k danému programu a také nemusí program přeložit vždy úplně správně. Také vznikají problémy s odinstalací programů takto nainstalovaných a velmi často dochází ke kolizím s balíčkovacím systémem. Kompilace také zabere množství času. Při překladu zdrojových kódů do binární podoby dochází k vysokému vytížení systému. U jednoduších aplikací kompilace zabere vteřiny či minuty u složitějších a rozsáhlejších projektů jako je například KDE může kompilace zabrat deset a více hodin. Samozřejmě záleží na výkonosti vašeho hardware. Na druhou stranu má ale instalace ze zdrojových kódů několik zásadních výhod. Především jde o teoretické zrychlení. Program, který je přeložen přímo pro váš typ hardware by měl běžet (a většinou běží) o něco rychleji. Teoreticky by se měla i zvýšit stabilita.

O překladu ze zdrojových souborů vyšel seriál Nebojíme se kompilace.

Obecně platí při kompilaci následují postup aneb svatá trojice:

```
./configure                  # můžou se připojit další volby jako např. --
prefix=/usr/local
make
make install                 # jako superuživatel
```

Doporučuje se místo závěrečného `make install` použít raději `checkinstall`. Nedojde tak k přímé integraci programu do systému bez evidence, což by jinak mohlo být při odinstalaci problematické (pokud součástí programu není `make uninstall`). Při použití `checkinstall` se nejdříve vytvoří nový balík, který je poté standartně dále naistalován balíčkovacím systémem. Tak předejdeme výše uvedeným problémům.

Instalace z balíčku

Pohodlnější způsob instalace je pomocí balíčků - jde o již (většinou) zkompilovanou aplikaci, zabalenou do jednoho souboru. Některé balíčkovací systémy řeší i závislosti aplikace - to znamená, že stáhnou z internetu (či si jinak vyžádají) ostatní programy a knihovny, které aplikace ke svému běhu potřebuje. Instalace je většinou velmi snadná. Toto je většinou nejpoužívanější způsob instalace nových programů.

V současnosti se nejčastěji používají tyto balíčkovací systémy:

apt

Balíčkovací systém distribuce Debian. Mimo jiné ho používá i velmi oblíbená distribuce Ubuntu. Balíčky mají koncovku `.deb`. Debian se pyšní obrovským množstvím balíčků, je tedy velmi pravděpodobné, že v repozitářích najdete vše potřebné. Balíčky obsahují buďto zkompilovaný program, takzvanou „binárku", anebo zdrojový kód (máte na výběr). Při správě balíčku si vystačíte s příkazem `apt-get`, ale mnohem snadnější je použít jeho grafický frontend Synaptic. Pomocí dalších nástrojů, jako např. alien lehce převedete .rpm balíčky na .deb

`apt-get update`	Aktualizuje seznam balíčků
`apt-get install balicek`	Nainstaluje do systému zadaný balíček (balíčky)
`apt-get remove balicek`	Odebere ze systému zadaný balíček (balíčky)
`apt-get upgrade`	Aktualizuje všechny balíčky nainstalované v systému
`apt-cache search vyraz`	Vyhledá balíčky související s daným výrazem
`apt-get dist-upgrade`	Upgraduje distribuci(najde aktualizace balíčků)

České APT-HOWTO
Hledání balíčků na webu Debianu

RPM

Velice známý, a hodně používaný systém balíčků. RPM znamená „RPM Package Manager" (dříve „Red Hat Package Manager" — *Redhatovský manažer balíčků*) a kromě samotného Red Hatu se používá především v distribucích jako je Mandriva, Fedora Core nebo Suse. Tudíž je pravděpodobnější, že se s ním jakožto začátečník setkáte. Samotný program `rpm` neinstaluje závislosti automaticky. Proto byla vyvinuta spousta nadstaveb:

- `urpmi` - v distribuci Mandrake/Mandriva
- `yum` - pro distribuci Fedora Core
- `yast` - pro distribuci Suse
- `apt4rpm` - port programu z Debianu pro systémy založené na rpm

Vyhledávače RPM balíků rpmfind.net a rpm.pbone.net

tgz

Tgz se používá ve Slackware. Je to tar.gz archív s určenou vnitřní strukturou. Slackwarovský systém balíčků však narozdíl od ostatních neumí vyřešit závislosti, a i samotná distribuce se začátečníkovi může jevit jako „hardcore". Programy pro správu balíčků se nazývají `pkgtools` a jsou to shellové skripty. Závislosti dokáže částečně řešit swaret, bohužel je ale stále málo balíčků které toto umožňují.

Hledání balíčků pro Slackware

Portage

Portage je balíčkovací systém distribuce Linuxu Gentoo, který je založen na systému portů z BSD. Tento systém balíčků většinu software kompiluje ze zdrojových kódů, podle nastavení v souboru `/etc/make.conf`. Ovládá se pomocí příkazu emerge, například `emerge glibc` nainstaluje balíček glibc.

pacman

Tento systém byl původně vyvinut pro distribuci Arch Linux, nyní jej používá ještě Rubix a Frugalware. Pacman umí všechny klasické věci, jako hlídání závislostí, instalace, odinstalování, apod. Součástí *pacmana* je i utilita *abs*, která skrze CVS udržuje strom PKGBUILD souborů, které slouží pro vygenerování balíčku ze zdrojových kódů. Arch Linux má rozsáhlý repozitář komunitních balíčků a každý uživatel může další balíčky přidávat právě zařazením svého PKGBUILDu do tohoto repozitáře. Tento systém rovněž umožnuje kompilaci celého systému ze zdrojových kódů, podobně jako Gentoo. Gentoo/Portage je však primárně zdrojová distribuce, zatímco Arch/Pacman je primárně binární.

Text

Text je přirozený, ve všech systémech prakticky stejný a v čase neměný způsob komunikace. Rozumějí mu lidé, není těžké ho naučit počítače (pokud je slušně formátovaný). Právě toto jsou důvody, proč je používán v unixových systémech k uchovávání konfigurace, komunikaci mezi programy, ať lokálně nebo po síti, a komunikaci s uživatelem.

Pokud je nutné upravit nějaký text v souboru, používají se na to textové editory. Mezi nejstarší a nejdokonalejší textové editory se řadí **vi**, jeho novější následník **vim** a jeho odvěký soupeř **emacs**.

Vi se vyskytuje na všech unixových systémech, je nenáročný, rychlý a chová se všude prakticky stejně. **Vim** staví na geniálním ovládání editoru vi a rozšiřuje jeho schopnosti o vlastní skriptovací jazyk, vizuální režimy výběru, pluginy a mnoho dalších užitečných vlastností. Kvalitní návod pro začátky s vimem je k dispozici na adrese http://www.kit.vslib.cz/~satrapa/docs/vim/.

Někdy, vlastně celkem často, přijde vhod upravit velké množství textu, který má přesný, známý formát. Nebo se nehodí čekat na uživatele, případně je vhodné upravovat volbu v konfiguračním souboru podle aktuální nálady a dostupnosti např. sítě. Pro tyto případy existuje několik nástrojů. Mezi základní a celkem jednoduše použitelné se řadí **sed**.

Sed umí aplikovat regulární výraz na proud dat (standardní vstup a výstup) nebo soubor. Například:

```
ps aux | sed 's/honza/anicka/g'
```

nahradí Honzu za Aničku ve výpisu procesů a

```
sed '/^ao=/s/=.*$/=alsa/' -i ~/.mplayer/config
```

přenastaví mplayer, aby používal Alsu místo něčeho jiného (používám dvojici takovýchto příkazů k přepínání mezi používáním lokální zvukové karty notebooku a domácího počítače s běžícím esd).

Občas se hodí vyhledat nějaký text v jednom či více souborech nebo vykousat z logu jen zajímavé záznamy. K tomu slouží program **grep**. Například

```
grep -R '/etc/motd' /etc
```

najde, kde všude se vyskytuje zmínka o souboru '/etc/motd' v adresáři /etc.

Podrobnosti o programech sed a grep jsou v manuálových stránkách.

Text a grafické rozhraní

Protože se někomu může zdát úprava například konfiguračního souboru pomocí vimu těžká a přílíš „textová“, byly vyvinuty různé editory textu pro X. Uživateli KDE často používané `kwrite` a `kdedit` jsou toho hezkým příkladem. Tyto editory se hodí, pokud chcete opravit přesný soubor, například `/etc/fstab`. V KDE stačí zmáčknout `Alt + F2` a zadat `kdesu kwrite`. Kdesu se vás zeptá na rootovské heslo, a spustí kwrite s právy roota - ovšem můžete kdesu klidně vynechat, pokud nepotřebujete práva superuživatele. Pak je také jednodušší si editor najít v nabídce.

Bylo by ovšem chybné domívat se, že kwrite je vhodným nástrojem pro psaní např. výroční zprávy – pro rozsáhlé texty s formátováním je vhodný OpenOffice.org

Pokročilá práce s textem

Pokud byste chtěli formátovat text na opravdu špičkové úrovni, bude Vaší volbou (La)TeX. Není to textový editor, ale značkovací jazyk (podobně jako HTML) pro formátování textu. LaTeX sice není tak jednoduchý jako HTML, ale výsledky jsou toho hodny. Typický způsob práce s ním je: napsat text ve vimu, ještě ve vimu obohatit značkami LaTeXu a poté protáhnout přes kompilátor.

A pokud si myslíte, že toto je v praxi nepoužitelné, podívejte se na knihy, které máte doma. Pokud máte nějakou novější fantasy nebo sci-fi, s velkou pravděpodobností najdete v tiráži poznámku „Sazba provedena systémem TeX“.

Pokud přeci jen dáváte přednost úpravám sázeného textu v grafickém režimu, může být vaší volbou program Scribus.

X Window

Historie

Původní X vytvořilo MIT (Massachusetts Institute of Technology) v roce 1984. X11 se pak objevilo v září roku 1987, které vede X.Org Foundation.

Prvotní myšlenka je od pánů z MITU Jima Gettyse a Boba Scheiflera (1984). X prošel více vývojovými stádii (X9, X10) a jako X11 se nakonec objevil v mailing listech na internetu, kde byl jako free software dále vyvíjen.

Po značném úspěchu MIT chtěla od projektu odstoupit, ale distributoři požadovali držení aspoň zlomku akcií. V roce 1993 se X Consortium stalo následníkem MIT X Consortium a došlo k uvolnění, 16 května 1994, X11R6. V roce 1995 se stal spolu s Motif toolkitem standardním desktopovým prostředím pro Unix systémy.

Roku 1997 přešlo X Consortium pod správu Open Group (otevřené společnosti), které vydávalo verze až po X11R6.4 patch 3.

V roce 1999 projektovalo X.Org X11R6.5.1, vývoj ale ustával. XFree86 (vytvořené roku 1992 pro IBM servery Thomasem Roell a Markem W) obsahovalo mnoho technických inovacích vzatých od X Consortia, zatímco X.Org mělo širokou podporu hardware a využití v linuxu. Z těchto důvodů došlo ke spojení a XFree86 se stalo čestným členem X.Org.

Nastal rok 2003. Popularita Linuxu stoupala spolu s popularitou X.Org, ke kterému začali přecházet i stálí vývojáři XFree86, v němž nastal rozpor. Proto začali diskutovat nad reorganizaci a kontrolou otevřeného vývoje.

V únoru 2004 XFree86 vydala verzi 4.4 pod více vyhrazenou licenci, kterou mnoho projektů spolehajících se na X nepřijalo - čímž začaly licenční spory. Větu o možném šíření pod BSD licencí vidělo Free Software neslučitelným s GNU GPL.

V roce 2004 lidé z X.Org a Open Group začali kontrolovat doménové jméno x.org, což byla radikální změna kontroly X. V září 2004 vyšlo X11R6.8 s novými featurami včetně předběžné podpory průhledných oken a jiných vizuálních efektů, obrazových zvětšení a miniatur a 3D výbavy.

X je šířen jako free software pod licencí MIT a podobnými.

Koncepce

Základem architektury X11 je *X server*. X server je zařízení (dnes nejčastěji počítač s potřebným hardwarovým a softwarovým vybavením) disponující jednou nebo více obrazovkami (fyzickými nebo virtuálními) a vstupními zařízeními - obvykle klávesnicí a polohovacím zařízením (myš, tablet). Tyto prostředky dává X server k dispozici klientům, kteří je chtějí využívat. Pomocí protokolu X11 klientská aplikace posílá X serveru požadavky na vykreslování (základní grafická primitiva) a naopak od X serveru dostává upozornění na *události* (events), odpovídající podnětům vstupních zařízení (stisk klávesy na klávesnici, pohyb myši apod.).

Komunikace mezi X serverem probíhá buď prostřednictvím lokálního (unix domain) socketu nebo pomocí TCP spojení. Z pohledu klientské aplikace jsou tyto varianty zcela rovnocenné, jakoukoli grafickou aplikaci v unixových systémech lze proto používat beze změn buď přímo na témže počítači jako X server nebo vzdáleně. Jediná změna, kterou musíme provést, je předání informace o X serveru, který má aplikace používat, viz kapitola vzdálený přístup. Výjimku tvoří pouze aplikace, používající kvůli rychlosti rozšíření pro přímý přístup ke grafické kartě, např. některé přehrávače videa nebo hry.

Window manager

X sice operuje s pojmem okna, ale v pojetí X serveru je okno pouze obdélníková oblast obrazovky, kterou lze použít pro vykreslování a ke které se vztahují události vstupních zařízení. X server sice eviduje uspořádání překrývajících se oken (z-order), ale nezabývá se vykreslováním rámečků ani interakcí s uživatelem (klávesové zkratky a tlačítka pro přesun nebo změnu velikosti apod.). Tyto funkce obstarává samostatná aplikace, označovaná jako *window manager* (správce oken). Vůči X serveru vystupuje window manager jako obyčejná klientská aplikace, předává mu požadavky na vykreslování (a změny atributů oken), naopak od něj dostává informace o podnětech uživatele, na které má reagovat.

Stejně jako normální aplikace, ani window manager nemusí být spouštěn na témže počítači jako X server. To je zejména případ *X terminálů*. X terminál je zařízení, které je vlastně jednoduchým počítačem vykonávajícím funkci X serveru, na kterém ale nejsou spouštěny vlastní aplikace (ani window manager). Uživatel se pomocí takového terminálu přihlásí k vzdálenému počítači (obvykle v lokální síti) a window manager a další aplikace jsou spouštěny tam. Výhodou je, že X terminál nevyžaduje příliš výkonný počítač, většinou je realizován jako bezdiskový a často nemá ani aktivní chlazení, takže je potom velmi tichý.

Další výhodou modulární struktury systému X Window je skutečnost, že si uživatel může vybrat window manager podle svých preferencí. Na jedné straně je nabídka minimalistických window managerů, které implementují pouze základní funkce (přesun a změna velikosti oken, jednoduché

menu), např. `twm` nebo `mwm`. Na opačném konci stojí prostředí, která implementují mnoho dalších pokročilých funkcí a správa oken je pouze velmi malou částí jejich funkcionality; proto se pro ně používá spíše termín *desktop manager*, v Linuxu se nejčastěji setkáváme s KDE a Gnome. Za vysokou míru komfortu ale samozřejmě platíme podstatně vyššími nároky na paměť a čas procesoru.

Existují dokonce i aplikace, kde není třeba správu oken provádět vůbec nebo by dokonce byla na závadu, protože počítač slouží pouze k běhu jedné konkrétní aplikace. Příkladem jsou informační kiosky. V takovém případě není třeba spouštět window manager vůbec, stačí X server a aplikace.

Další komponenty

Další důležitou komponentou je *display manager*, starající se o autentizaci a autorizaci klientských aplikací. Povolíme-li totiž síťovou komunikaci aplikací s X serverem, povolili bychom tím přísup k X serveru komukoli, kdo je schopen navázat TCP spojení, a to by mohl být v krajním případě i celý Internet. Kdokoli by pak mohl nejen zobrazovat na náš X server, ale dokonce i získávat události vstupních zařízení, a tedy např. sledovat, co píšeme na klávesnici. Proto display manager omezuje přístup k X serveru pouze na důvěryhodné klienty, podrobnosti viz kapitola vzdálený přístup. Display manager je také zodpovědný za zobrazení a funkci úvodního přihlašovacího okna a případně i sdílení informací s dalšími display managery protokolem XDMCP.

Další užitečnou aplikací může být *font server*, který poskytuje X serverům fonty. Této možnosti se využívá zejména u hardwarových X terminálů, do nichž není možné další fonty instalovat obvyklým způsobem. Proto se fonty instalují na font server a terminálům se pouze v konfiguraci nastaví umístění font serveru, který mají využívat. Součástí instalace linuxového X serveru je i jednoduchý font server (program `xfs`).

Principy práce v X

Kopírování

Existují dva způsoby kopírování. Prvním je použití kombinace `Ctrl+C` a `Ctrl+V`, nebo odpovídající položky kontextového menu (vyvolaného pravým tlačítkem). Druhý, o poznání zajímavější spočívá v použití středního tlačítka myši. Prostě označíte text, a pak se přesunete na místo, kam ho chcete vložit, a kliknete tam středním tlačítkem. Tyto dva způsoby jsou zcela nezávislé, takže můžete dvě různé věci ve stejnou chvíli ve schránce. Použijete-li `xclipboard` můžete jich tam samozřejmě mít i víc. Druhý způsob se hlavně hodí pro nativní consoli X, `xterm`. Pokud jste hrdým majitelem myši s méně než třemi tlačítky, nezoufejte. X server umí emulovat třetí tlačítko. Emulace se provede při synchronním stisku obou tlačítek.

Sejmutí obrazovky

Sejmutím se v tomto případě myslí provedení screenshotu, nikoli zabíjení. Pro to máme programy `xwd` a `xwdtopnm`. `Xwd` nám típne vybrané okno a výsledek uloží ve formátu X11 dump file, který pomocí `xwdtopnm` převedeme na PPM (portable pixmap).

Běžné použití:

`xwd | xwdtopnm > nazev_souboru_pro_tipanec` - típne okno na které kliknete.

`xwd -nobdrs | xwdtopnm > nazev_souboru_pro_tipanec` - típne okno na které klinete, ale vynechá jeho okraje

`xwd -root | xwdtopnm > nazev_souboru_pro_tipanec` - típne celou obrazovku (takzvané kořenové okno - root window)

Otevření terminálu

Pokud potřebujete terminál (konzoli se shellem) v grafickém rozhraní, pak stačí spustit `xterm`. Zatoužíte-li přerušit aplikaci spuštěnou v `xterm`u, lze to provést zmáčknutím `Ctrl + C`.

Zatuhlé aplikace

Pro ty ojedinělé případy zatuhnutí aplikací v X je zde `xkill`. Tento malý, ale šikovný prográmek změní kurzor v zaměřovač. Pak stačí kliknout na okno zatuhlé aplikace. Častou chybou začátečníků je použití `xkill` na panel, ve snaze ukončit aplikaci, která je na panelu nějak reprezentována. V případě, že spustíte xkill omylem, nebo si rozmyslíte zabíjení aplikace, použijte pravé tlačítko - vypne se a kurzor získá zpět normální tvar. **Ale pozor**, xkillem jste zabili jen grafické rozhraní. Musíte tedy použít `kill` nebo `killall`, aby jste dokonali očistu.

Spouštění

Standardní spouštění X serveru z příkazového řádku se spouští příkazem `startx`, ať již se jedná o účet obyčejného uživatele nebo superuživatele.

Pokud X na počítači již běžel, ale byl vypnut, jeho opětovné spuštění provedeme jako root: `init 5`. Na Debianu, Ubuntu a odvozených požijete `/etc/init.d/kdm start`, pokud používáte KDM, nebo `/etc/init.d/gdm start` pro GDM.
Nebo také spustíme přímo GDM (Gnome) či KDM (KDE) `gdm` nebo `kdm`.

Zastavení (vypnutí)

Pokud se dostaneme do situace, že potřebujeme X vypnout, nepoužijeme `killall`. Místo toho (jako root) zadáme `init 3`. Pokud používáte Debian, zadáte `/etc/init.d/gdm stop` nebo `/etc/init.d/kdm stop`. Ve všech případech, kromě `killall` dojde ke správnému vypnutí X, včetně uzavření všech grafických aplikací.

Bez managerů to nepůjde

Desktop manager je sada programů, které stojí nad X servrem a činí náš desktop (grafické rozhraní) použitelným. Zajišťují nám pozadí, ikony na něm (neboli na ploše), panely, virtuální plochy a okraje oken. Především vykreslování *okrajů* oken je jejich důležitou funkcí. Bez nich bychom byli "nahraní".

Window manager zajišťuje pouze okraje oken, občasně i panel a pozadí.

Desktop managery

KDE

V roce 1996 byl založen projekt dnes asi nejpoužívanějšího desktop manageru KDE (K Desktop Environment). Je stále aktivně vyvíjen, nejnovější vydání je 3.5. Některé distribuce staví KDE jako svůj defaultní desktop manager. KDE je napsáno v C++ pomocí knihoven Qt. Ačkoli někteří říkají, že se příliš podobá MS Windows a že není hezký, opak je pravdou. KDE ve stavu, v jakém je dostanete po instalaci může vyžadovat sice trochu péče, ale výsledek rozhodně stojí za to. Ačkoli je KDE vybavené množstvím programů a grafických elementů, není na škodu navštívit kde-look.org, pro věci týkající se vzhledu (například dekorace oken); a kde-apps.org, který nabízí roztodivné aplikace. Nutno podotknout, že materiál ze dvou výše zmíněných webů nepodléhá nijak týmu vývojářů KDE a dokonce ani nemusí být GPL či ani svobodný software.

Gnome

Grafické rozhraní GNOME je postaveno na knihovnách GTK. GNOME samotné se může zdát nezáživné a těžkopádné, ale mezi aplikacemi psanými v GTK jsou opravdové špičky. Jeden z hlavních problému GNOME je, že jeho vývojáři se domnívají, že uživatelé jsou málem negramotní.

XFCE

Pokud chcete hezký a funkční desktop při nízké spotřebě systémových zdrojů, tak XFCE je přesně pro vás. Téměř každá část XFCE je šířena pod jinou licencí, většinu tvoří GPL, BSD a X11 licence.

Window managery

Fluxbox

Před nějakým časem se část vývojářů Blackboxu odštěpila a šla svou vlastní cestou. Vytvořili fork zvaný Fluxbox. Jedná se sice o lehký, ale přesto však příjemný správce oken. Pro své nízké hardwarové nároky je dobrý vhodný pro používání na starších strojích (Pentia okolo 150 MHz a 32 MB RAM). Mezi hlavní přednosti Fluxboxu patří až neuvěřitelně vysoká konfigurovatelnost. Dále možnost vkládání oken do sebe - libovolná lze okna sloučit do sebe a přepínat mezi nimi pomocí záložek. Fluxbox má svůj panel, podporu virtuálních ploch a také podporu dockapps známých z Window Makeru. Jeho záporem je absence grafického konfigurátoru

jako má KDE - nastavení se tedy musí provádět editací textových souborů.

Blackbox

Jak bylo již řečeno, Blackbox je "rodičem" Fluxboxu - proto spolu sdílí většinu vlastností a jejich styly a témata jsou navzájem 100% kompatibilní. I pro lehce pokročilého uživatele může být šokující absence panelu s nabídkou a hodinami. Ale nic není ztraceno - stačí se podívat na Blackbox Addons (addons znamená "přídavky"). Tam budete muset sáhnout i pro podporu ikon na ploše, pokud by Vám k něčemu snad byla. Na závěr by bylo vhodné zmínit, že Blackbox je distribuován pod licencí MIT (Masachusets Instute of Technology).

Metacity

Metacity je windowmanager vyvinutý uživateli RedHatu. FIXME

Aplikace

Klasické X aplikace

V rámci X bylo vytvořeno několik aplikací, které se staly téměř slavnými a dnes jsou předělány pro např. KDE. Mezi takové známé prográmky patří kupříkladu `xeyes`, `xcalc`, `xwd` a `xkill`.

Aplikace KDE

Jak jsem již předeslal, KDE obsahuje velké množství aplikací. Mezi nepochybně zajímavé patří Kicker, což je vlastně panel na dolním okraji obrazovky. Můžete mít takovýchto panelů několik a různě rozmístěných po okrajích. Vše, co se na panelu nachází, můžete přesouvat a vytvořit si vzhled přesně dle vašeho přání. KDE mimo jiné obsahuje i svůj vlastní kancelářský balík KOficce. Ten je sice mnohem mladší něž samotné KDE, ale přesto je velice dobrý. Další KDE aplikace hodná pozornosti je SuperKaramba, která od KDE 3.5 standardně obsažena v instalaci. Aplikace samotná nic nedělá, potřebujete widgety . Ty vám mohou zobrazovat stav počítače, počasí či usnadnit spoustu aktivit. No a dále třeba Konqueror - jeden z nejlepších prohlížečů vůbec s vykreslovacím jádrem KHTML. Zvládá vektorovou grafiku (KSVG) a dokonce už podporuje i některé prvky z CSS3. Od verze 3.5 korektně zobrazuje Acid2 test.

Aplikace "nikoho"

Všechny aplikace nemusejí nutně využívat nějakou knihovnu a být součástí něčeho většího. Podívejme se například na Blender (3D modelování), vidíte že má velice svébytný vzhled. Nenechte se zmást tím, jak vypadá, v každém případě potřebuje X.

Terminál v grafickém rozhraní?

Když potřebujete shell a jste právě v grafickém rozhraní, není nic pohodlnějšího, než spustit `xterm` nebo `konsole` či `gnome-terminal` popřípadě `yakuake`, voila - můžete používat váš oblíbený shell a přitom sledovat film.

Formulářové toolkity

Formulářové (tlačítkové) aplikace využívají k vykreslení svého gui (rozhraní) formulářových toolkitů. Mezi nejznámější patří GTK a QT.

Qt

Aplikace napsané v Qt zapadnou do výše popisovaného prostředí KDE, které v něm je taktéž napsáno. Jako programovací jazyk se využívá především C++ (Qt je portované i do jiných jazyků, jako například Python (PyQt). Qt obsahuje širokou škálu funkcí a po zavedení jeho knihoven do paměti je i svižné. Problém nastává s licencemi, jelikož programy napsané v Qt nemají oficiální port na jiné platformy (například Windows). S příchodem Qt4 by se situace měla zlepšit.

GTK (gimp the toolkit)

V GTK je naopak zase napsané celé Gnome a mnoho multiplatformních aplikací. Jako programátorský jazyk je možné použít C. GTK nezabere v paměti zpočátku tolik místa, ale se svižností je na tom (IMHO) o trochu hůře. Zase jeho největší výhodou je výše zmiňovaná přenositelnost. Mezi zástupce aplikací můžeme uvést internetový prohlížeč Firefox, poštovní program Thunderbird a hlavně grafický editor Gimp, pro který byl toolkit původně napsán.

"Herní" aplikace a knihovny

Existuje skupina programů jako jsou hry, různé screensavery a nástroje pro 3D modelování jako je např.Blender. Ty obvykle využívají 2D či 3D knihoven pro práci s grafickými funkcemi a akcelerací grafické karty.

Nejznámějšími je grafická knihovna SDL pro práci s 2D grafikou a mezi čistě v ní napsáné aplikace můžeme uvést například strategii Wesnoth či 'sestřelovačku' kuliček Frozen Bubble.

Open GL se využívá pro 3D aplikace (jeho alternativou je na Windows platformě DirectX). Příklady her napsaných v Open GL: závod tučňáka TuxRacer, střílečka Cube či komerční hry například od ID software jako Doom III, Wolfstein nebo Quake.

Vzdálený přístup

Protokol X11 je navržen tak, aby ho bylo možné používat nejen tehdy, když X server a klientská aplikace běží na témže počítači, ale i v případě, že běží na různých počítačích, které jsou propojeny sítí. V takovém případě je ke komunikaci využíváno TCP spojení. Aby mohla aplikace komunikovat s X serverem, je třeba jí sdělit, na který X server (a případně na kterou obrazovku) má zobrazovat. Tato informace je obvykle předávána prostřednictvím proměnné prostředí `DISPLAY`, klasické grafické programy podporují i parametr příkazové řádky `-display` (nebo zkráceně `-disp`).

Hodnota proměnné (resp. argument) má obecně tvar '*host*`:`*display*`.`*screen*'. Nepovinný parametr *host* definuje počítač, kde se X server nachází. Většinou obsahuje jméno počítače nebo IP (IPv4 nebo IPv6) adresu. Není-li uveden, kontaktuje se X server na stejném počítači, většinou ale ne pomocí TCP spojení přes lokální smyčku, ale přes lokální (unix domain) socket.

Parametr *display* určuje číslo konkrétního X serveru na zvoleném počítači. Na jednom počítači může totiž X serverů běžet více, například je-li současně otevřeno více grafických konzolí. Jiným příkladem je spuštění programu `Xnest`, který implementuje X server používající jako virtuální obrazovku okno zobrazované na jiném X serveru. Obvyklé číslování X serverů je od nuly, nejčastěji se zde proto setkáme s hodnotou `0`. Tento parametr je také (včetně dvojtečky) jediný povinný, minimální (a také nejčastější) specifikace obrazovky je tedy '`:0`'. Z praktického hlediska parametr *display* při TCP komunikaci určuje port, na nějž je navazováno spojení; jedná se o port 6000+*display*.

Poslední (opět nepovinný) parametr *screen* určuje obrazovku, na kterou má aplikace zobrazovat. Obrazovky (má-li jich X server více) se číslují opět od nuly (nula je i defaultní hodnotou). Číslo obrazovky na rozdíl od předchozích parametrů neovlivňuje navázané spojení, informace je používána interně v rámci komunikace protokolem X11. Obrazovky v tomto smyslu nemají nic společného s virtuálními pracovními plochami, které nabízí většina window managerů. Nejčastěji se s více obrazovkami setkáváme u počítačů s více monitory, kde číslo obrazovky určuje (fyzický) monitor, na který bude aplikace zobrazovat.

Praktická použitelnost aplikace běžící na vzdáleném počítači a zobrazující na lokální X server je ovlivněna parametry síťového spojení. Je-li spojení realizováno lokální sítí, nebývá to problém; u 100 Mb/s ethernetu na chování většiny aplikací není vzdálená komunikace příliš znát. U pomalejších spojení (kromě přenosové rychlosti zde hraje roli i zpoždění paketů) jsou pochopitelně odezvy delší. Hodně zde záleží i na tom, co aplikace zobrazuje. Vykreslování jednoduchých objektů nebo vypisování textů není příliš náročné, ale např. při prohlížení fotografií je třeba přenášet celé bitmapy, což může být poměrně náročné. Nezanedbatelnou výhodou je i skutečnost, že tímto způsobem můžeme používat grafické aplikace i na počítači, kde není vůbec nainstalován X server, např. na výkonném aplikačním serveru, ke kterému přistupují uživatelé ze svých pracovních stanic. Nelze zapomínat ani na to, že protokol je univerzální: není tedy omezen na prostředí Linuxu, ale lze jej používat i mezi různými platformami.

Možnost zobrazování na X prostřednictvím TCP/IP sítí (a tedy i Internetu) je velmi užitečným nástrojem, jedná se ale o dvousečnou zbraň. Pokud bychom neaplikovali mechanismus, který by omezil potenciální klienty, mohl by kdokoli používat náš X server, a to nejen k zobrazování, ale i k získávání událostí od vstupních zařízení; mohl by tedy mimo jiné např. sledovat, co píšeme na klávesnici. Nepotřebujeme-li vůbec přistupovat na X server vzdáleně, může být vhodnější použít při jeho spouštění parametr '`-nolisten tcp`', který způsobí, že X server nebude vůbec poslouchat na TCP portu 6000+*display* a umožní pouze komunikaci lokálních klientů přes lokální socket. Některé novější distribuce naopak spouštějí X server s tímto parametrem defaultně, takže naopak

chceme-li komunikovat vzdáleně, je potřeba to explicitně povolit. V dalších částech se ale zaměříme na situaci, kdy vzdáleně komunikovat chceme, ale potřebujeme klienty autorizovat.

Seznam povolených adres

Nejjednodušší formou autorizace je rozlišení klientů podle IP adresy. Display manager udržuje seznam adres (standardně prázdný), z nichž je povolen přístup všem klientům bez dalších omezení. K manipulaci s tímto seznamem lze použít příkaz `xhost`, bez parametrů vypíše seznam povolených adres (standardně přeložených na DNS jména). Přidání položky do seznamu lze provést příkazem '`xhost +`*host*' (znak + lze vynechat), odebrání příkazem '`xhost -`*host*'.

Jak už to bývá, je tato varianta sice nejjednodušší, ale také nejméně vhodná. Jednak povolením adresy konkrétního počítače povolujeme přístup automaticky všem jeho uživatelům, jednak při podvržení IP adresy server nepozná, že se nejedná o oprávněného klienta. Pomineme-li tedy specifické případy typu lokální sítě o dvou počítačích a jednom uživateli. nelze tuto metodu v dnešní době pro praxi doporučit.

Autentizační cookies

Při spuštění seance (session) display manager generuje náhodný klíč (cookie), který je s touto seancí svázán. Klient, který se prokáže znalostí této cookie, má povolen přístup k X serveru, aniž by jeho IP adresa musela být v seznamu z předchozí sekce. Cookie je spolu s informací o serveru (zapisuje se ve stejném formátu jako hodnota proměnné `DISPLAY` s vynecháním čísla obrazovky) uložena do autentizační databáze, která je standardně v souboru `.Xauthority` v domácím adresáři přihlášeného uživatele. Z tohoto souboru si pak cookies berou jednotlivé grafické aplikace a používají je pro autentizaci vůči serveru.

S obsahem autentizační databáze lze pracovat příkazem `xauth`, a to buď interaktivně (spustíme-li ho bez parametrů) nebo neinteraktivně (předáme-li mu příkaz přímo jako argument). Základní příkazy jsou:

- `list` … zobrazení cookies z databáze
- `extract` *`file server`* … zkopírování cookie pro server *server* do souboru *file*
- `merge` *`file`* … přidání cookie ze souboru do databáze

Jako jméno souboru lze uvést i znak '`-`', cookie se pak extrahuje na standardní výstup a načítá se ze standardního vstupu. Pokud v takovém případě nepřesměrováváme výstup nebo nepoužijeme rouru, je vhodnější použít příkazy `nextract` a `nmerge`, používající formát složený pouze z číslic, který lze zobrazovat na terminál a kopírovat pomocí selection nebo copy&paste.

Přihlásíme-li se tedy na vzdálený počítač např. pomocí SSH a chceme-li na něm spouštět grafické aplikace tak, aby zobrazování probíhalo na náš lokální X server, je třeba nastavit odpovídajícím způsobem proměnnou `DISPLAY` tamnímu shellu (nesmíme ale zapomínat, že shell a posléze i grafická aplikace poběží na vzdáleném počítači, nelze tedy psát `localhost` nebo jméno počítače vynechat). Dále ale také musíme zkopírovat autentizační cookie z lokální databáze do vzdálené. To lze provést buď extrakcí do souboru, jeho překopírováním a importem cookie do databáze, nebo extrakcí na terminál a překopírováním do druhého okna, kde jsme přihlášeni na vzdáleném stroji a předem jsme použili '`xauth nmerge -`'.

Transparentní tunelování pomocí SSH

Zásadní nevýhodou předchozích postupů je skutečnost, že řeší pouze otázku autentizace a autorizace (a ani to ne zcela spolehlivě), ale žádným způsobem nechrání data, která si mezi sebou

vyměňují X server a klientská aplikace, a to proti odposlechu ani proti podvržení. Pokud tedy komunikace mezi oběma počítači není zabezpečena např. pomocí IPsec, není takové řešení příliš vhodné, a to zejména, komunikují-li mezi sebou přes Internet. Protože X11 komunikace používá jedno TCP spojení, je ideálním kandidátem na zabezpečení pomocí SSL (např. programem stunnel). Protože se ale dnes na vzdálený počítač, kde aplikaci spouštíme, většinou připojujeme pomocí SSH, je jednodušší využít možnosti transparentního tunelování X11 přes jeho zabezpečený přenosový kanál.

K pochopení toho, jak funguje tunel, je třeba mít na paměti, že X server je na straně, kde je SSH klient, a naopak. Tunelování není třeba nijak složitě nastavovat, pouze je nutné, aby tato funkce byla povolena v konfiguraci klienta (`ssh`) i serveru (`sshd`) a aby byl klient spouštěn s parametrem `-X` a měl nastavenou proměnnou `DISPLAY` na lokální X server. Je-li tomu tak, SSH démon vytvoří virtuální X server (standardně s číslem od 10 výše, aby nedocházelo ke kolizi s případnými lokálními X servery na jeho straně. Na tento X server (tj. typicky na hodnotu '`:10`') pak také nastaví shellu (nebo jinému spouštěnému programu) proměnnou `DISPLAY`.

Spuštěná grafická aplikace tedy veškeré požadavky posílá na virtuální X server, ten je ale pouze přeposílá (zabezpečeným spojením) SSH klientovi, který je dále předá lokálnímu X serveru na straně uživatele. Podobně, pouze v opačném směru, jsou posílány odpovědi X serveru aplikaci. Aplikace samozřejmě nemusí běžet na stejném počítači jako SSH démon, podobně SSH klient nemusí běžet na stejném počítači jako X server. Pak je ale zabezpečený jen úsek mezi SSH démonem a SSH klientem, to ale může být dostatečné, např. jsou-li koncové úseky realizovány relativně bezpečnou lokální sítí, zatímco tunelovaný úsek vede přes Internet.

Kromě obvyklých bezpečnostních opatření při používání SSH (zejména kontrola pravosti veřejného klíče serveru při prvním přihlášení) nesmíme zapomínat, že tunel chrání jen před útoky "po cestě". Není tedy ochranou před lokálním útokem na koncových počítačích. Zejména pokud bychom nemohli důvěřovat uživateli `root` na vzdáleném počítači, zpřístupňujeme mu otevřeným tunelem svůj lokální X server (po dobu, kdy je tunel navázán). Nesmíme také zapomínat, že při větších datových tocích šifrování a autentizace dat zatěžuje procesor obou stran. Naopak, u pomalých spojení lze zapnutím komprese (přepínač `-C`) přenos urychlit snížením objemu přenášených dat.

Lokální komunikace pod jiným uživatelem

Problematika lokální komunikace do této kapitoly zdánlivě nepatří, ale vše, co bylo řečeno v předchozích sekcích, se vztahuje i na aplikace běžící na témže počítači jako X server. Procesy uživatele, který se přihlásil do grafického prostředí, mají samozřejmě přístup k vygenerované cookie, takže jim je přístup umožněn. Pokud ale použijeme příkaz `su`, nebudou mít procesy spouštěné pod cílovým uživatelem k X serveru povolen přístup.

Povolení přístupu pomocí '`xhost localhost`' není vhodným řešením, protože tím bychom povolili přístup *všem* lokálním uživatelům a procesům. Řešení pomocí SSH je zase zbytečně náročné, protože šifrování a autentizace jsou při lokální komunikaci zcela zbytečné a pouze by zatěžovaly procesor. Nejvhodnějším řešením je tedy zkopírování autentizační cookie do databáze cílového uživatele. Lze to provést stejným způsobem jako při vzdálené komunikaci, kopírování je zde ale o něco jednodušší. Je-li cílovým uživatelem `root`, lze navíc využít toho, že má povoleno čtení jakéhokoli souboru ve filesystému a nechat ho extrahovat cookie přímo z databáze přihlášeného uživatele:

```
xauth -f ~user/.Xauthority extract - :0 | xauth merge -
```

(`user` reprezentuje jméno přihlášeného uživatele).

Jednodušší alternativou je použití příkazu `sux`, který funguje podobně jako `su`, ale navíc

automaticky kopíruje autentizační cookie od původního uživatele k novému. Také lze použít autentizační modul `pam_xauth`, který způsobí automatické předání autentizační cookie při použití příkazu `su`. Stačí přidat do souboru `/etc/pam.d/su` řádek

```
session  optional       pam_xauth.so
```

Pomocí konfiguračních souborů `~/.xauth/export` a `~/.xauth/import` lze pak definovat, kterým uživatelům se mají autentizační cookies předávat (neexistuje-li soubor `export`, pak normální uživatel předává všem, `root` nikomu) resp. od koho se mají přijímat (neexistuje-li soubor `import`, přijímá se od všech).

XDMCP - Linux jako terminál

Protože window manager je z pohledu X serveru klientská aplikace jako každá jiná, může samozřejmě i on běžet na jiném počítači než X server. Můžeme pochopitelně spouštět i window manager se zobrazováním na jiný X server pomocí proměnné `DISPLAY` nebo parametru `-display`, např.:

```
Xnest -geometry 1024x768 :1 &
fvwm -display :1 &
```

V praxi je ale vhodnější ponechat proces přihlášení, otevření seance a případnou volbu window manageru (a další konfigurace) na display manageru (`xdm`, `kdm`, `gdm`, `wdm`) počítače, ke kterému se přihlašujeme. Počítač, u něhož uživatel sedí, je pak vlastně degradován do role klasického X terminálu, na němž se zobrazí přihlašovací obrazovka vzdáleného počítače. Komunikace mezi X serverem a vzdáleným display managerem je realizována protokolem XDMCP (X Display Manager Control Protocol), standardně je pro něj používán UDP port 177.

Chceme-li se takto přihlásit ke vzdálenému počítači, je třeba při spouštění X serveru tento počítač specifikovat parametrem `-query` (argumentem je jméno nebo IP adresa):

```
X -query 192.168.2.65
```

Místo lokálního přihlašovacího okna se objeví přihlašovací okno nebo obrazovka vzdáleného display manageru. Po přihlášení je spuštěn (na vzdáleném počítači) odpovídající window manager podle tamní konfigurace.

Chceme-li, aby náš počítač fungoval jako terminál pro více vzdálených stanic, můžeme spustit display manager (konkrétní nastavení závisí na volbě display manageru) v režimu, kdy je uživateli místo lokálního přihlašovacího dialogu nejprve zobrazen seznam počítačů, na které se může přihlásit, a teprvé poté, co si ze seznamu vybere cílovou stanici, je zobrazen její přihlašovací dialog (obrazovka). Seznam může být buď definován staticky v konfiguraci lokálního display manageru nebo může být vytvářen dynamicky. Ve druhém případě display manager rozešle broadcast datagram s dotazem, kdo je ochoten povolit přihlášení, a zobrazí se seznam stanic, které odpověděly. Tento seznam se pak obvykle periodicky aktualizuje. Je možná i kombinace staticky konfigurovaného a dynamicky generovaného seznamu.

XDMCP protokol neobsahuje prakticky žádné bezpečenostní prvky a vzhledem k použití UDP není ani jeho dodatečné zabepečení příliš snadné. Jeho použitelnost je proto omezena prakticky jen na důvěryhodné lokální sítě, kde se používá pro klasické X terminály a tenké klienty. Samozřejmě i použití XDMCP podléhá konfiguraci autorizačních pravidel, ale jejich výklad by byl nad rámec tohoto letmého seznámení. Bližší informace lze získat např. z těchto zdrojů:

- Domácí síť - III
- FreeBSD v malej firme - 6 (terminálové služby)
- Linux XDMCP HowTo

Síť

Síťové protokoly

Rodina protokolů TCP/IP, kterou dnes používáme k realizaci naprosté většiny síťové komunikace, byla navržena na přelomu 70. a 80. let s cílem vytvořit robustnější model síťové komunikace, který by byl schopen se do určité míry vypořádat i s výpadky částí sítě. Základní (a v té době celkem revoluční) myšlenkou je *packet switching* (přepínání paketů): data nejsou posílána jako souvislý proud (stream), ale po samostatných blocích (packet), a jednotlivé uzly sítě samy rozhodují, kudy budou pakety dále posílat.

V praxi je tato myšlenka realizována sadou protokolů, implementujících potřebné funkce. Protokoly obvykle rozdělujeme do několika úrovní (vrstev). Místo abstraktního ISO/OSI modelu, který pracuje se sedmi vrstvami, při výkladu TCP/IP většinou používáme zjednodušený pětivrstvý model (některé protokoly v TCP/IP modelu zastávají funkci více vrstev ISO/OSI modelu).

Vrstvám odpovídají ve struktuře paketu hlavičky jednotlivých protokolů. Základem paketu je blok aplikačních dat (podle okolností může mít ale i nulovou délku), před který postupně jednotlivé protokoly přidávají (v pořadí shora dolů) hlavičky se svými informacemi. Tyto hlavičky jsou pak moduly příslušných protokolů v opačném pořadí odebírány na straně příjemce.

Nejvýše je *aplikační vrstva*, tak označujeme data aplikačních protokolů jednotlivých síťových služeb. Takových protokolů existuje obrovské množství, z nejznámějších uveďme např. HTTP, SMTP, FTP, NTP. Z pohledu TCP/IP se jedná o data, která je třeba přenést k cílovému příjemci. O to se starají nižší vrstvy.

Nejníže je v modelu vrstva *fyzická*. Na rozdíl od vyšších vrstev se nejedná o softwarovou vrstvu (protokol), tímto označením rozumíme konkrétní fyzické médium, které používáme k přenosu dat. Příkladem může být např. twisted pair kabeláž ve většině lokálních ethernetových sítí, koaxiální kabel, optické vlákno nebo telefonní linka. Médium ale nemusí být hmotné - např. v případě bezdrátových sítí v mikrovlnném pásmu (wi-fi, breezenet) nebo optických pojítek.

Nejníže ze softwarových vrstev je *linková vrstva*. Jedná se o nejnižší komunikační protokol, sloužící k přenášení dat po fyzickém médiu. Tento protokol je většinou úzce svázán s konkrétní volbou média, ale tato korespondence nemusí být 1:1, např. ethernet bývá v praxi implementován nejen na twisted-pair kabeláži, ale můžeme se setkat s jeho implementacemi pomocí koaxiálního kabelu nebo naopak optických vláken. Jiným příkladem protokolu linkové vrstvy je PPP, protokol používaný k realizaci vytáčeného připojení (dial-up) nebo propojení počítačů přes sériovou linku. Podstatnou vlastností protokolů linkové vrstvy je skutečnost, že řeší pouze komunikaci mezi uzly, které jsou *přímo* spojeny (odtud i název).

Globální adresaci a směrování má na starosti vrstva *síťová*, v praxi realizovaná téměř výhradně protokolem IP (vyskytujícím se ve dvou verzích, IPv4 a IPv6). Zatímco i u protokolů linkové vrstvy existují adresy a např. v případě ethernetových MAC adres jsou (nebo by aspoň měly být) dokonce globálně jednoznačné, nelze je použít ke směrování paketů, protože z takových adres nelze poznat, kde cíl hledat. Adresy protokolu IP (IP adresy) jsou ale přidělovány hierarchicky tak, že delegace jednotlivých rozsahů odpovídá topologii sítě. Z cílové IP adresy lze proto určit, kudy máme paket

dále poslat, tedy alespoň následujícího prostředníka (hop) po cestě. Kromě této své základní funkce řeší protokol IP ještě některé další, např. fragmentaci (rozdělení příliš dlouhých paketů na několik kratších) nebo označení paketů podle typu provozu (ToS - type of service).

Některé důležité problémy ale protokol IP záměrně neřeší. Například adresování v IP hlavičce je pouze na úroveň konkrétního uzlu sítě, ale neumožňuje adresovat konkrétní proces. Nelze tak (podle IP hlavičky) např. rozlišit paket určený webovému serveru od paketu pro SMTP klienta. Dalším problémem může být skutečnost, že jednotlivé IP pakety jsou posílány a zpracovávány zcela samostatně a nelze poznat, zda dva pakety patří do téhož datového toku. Protokol IP také neřeší otázku ztracených paketů (nelze detekovat, zda se některé pakety neztratily) ani pořadí paketů (pakety mohou do cíle dorazit v opačném pořadí, než v jakém byly odeslány). U některých typů komunikace nemusí být tato skutečnost na závadu, tam, kde ano, musíme tyto problémy řešit na úrovni vrstvy transportní a aplikační.

Nejpoužívanějšími protokoly transportní vrstvy jsou dnes UDP a TCP. UDP (User Datagram Protocol) lze chápat jako minimalistický transportní protokol, zavádějící pouze pojem portu, který lze chápat jako adresu konkrétního procesu (přesněji socketu) v rámci cílového uzlu. UDP je ale stále bezstavový protokol (nelze tady mluvit v pravém slova smyslu o spojení), neřeší otázku ztracených paketů ani jejich pořadí. Přesto se často používá pro svou jednoduchost a nižší režii, a to zejména tam, kde tyto otázky řešit nepotřebujeme.

Opačný přístup je reprezentován protokolem TCP (Transmission Control Protocol). Ten naopak zavádí *spojení* mezi dvěma porty na koncových uzlech. Z pohledu klientské aplikace se takové spojení chová podobně jako roura pro komunikaci mezi dvěma procesy (ale na rozdíl od roury je TCP spojení oboustranné), je zaručeno, že posloupnost bytů (stream), kterou jedna strana odešle, dostane ve stejné podobě druhá strana. Protokol TCP se stará o detekci a opakované odeslání ztracených dat, stejně jako o přerovnání dat z paketů, které dojdou ve špatném pořadí. TCP tak poskytuje aplikační vrstvě poměrně vysokou míru komfortu, většina komunikace proto dnes používá jako transportní protokol TCP. Nevýhodou ale může být vyšší režie a relativně pomalá reakce na výpadky, proto se pro některé účely (např. většina DNS dotazů nebo VoIP) dává přednost UDP.

Ze struktury vrstev se trochu vymyká protokol ICMP (Internet Control Message Protocol). Pakety (message) tohoto protokolu jsou přenášeny přímo prostřednictvím IP protokolu, ICMP ale nelze považovat za transportní protokol, protože neslouží k přenášení aplikačních dat. Tento protokol slouží k diagnostickým a servisním účelům. Příkladem aplikací ICMP jsou zprávy o nedoručitelnosti paketu (destination unreachable), pakety generované příkazem `echo` (ICMP echo a echo reply) nebo některé servisní typy zpráv (redirect).

Maximální (teoretická) velikost IP paketu je 65535 B, ale limitujícím faktorem je většinou linková vrstva. Protože většina paketů aspoň jednou projde přes ethernet (nebo jeho ekvivalent), bývá většinou velikost paketů volena podle jeho limitu (1536 B), odtud nejobvyklejší hodnota 1500 B. To je ale samozřejmě pouze maximální hodnota, pakety často bývají i výrazně kratší, zejména u interaktivních aplikací. Hlavičky IP a TCP mají velikost 20-60 B (obvyklejší jsou hodnoty u dolní hranice), UDP a ICMP hlavičky mají 8 B, ethernetová hlavička 14 B (navíc 2 B na konci paketu kontrolní součet).

Historie

Projekt *Netfilter/IPtables* založil roku 1998 sympatický Rusty Russel, který je také autorem předchůdce *IPtables* - projektu *IPchains*. Rusty je také zakladatelem týmu vývojářů (tzv. *Netfilter Core Team*), kteří se o celý rozsáhlý projekt starají. Projekt se postupem času z prostého paketového filtru obdařeného jen základními vlastnostmi rozrostl do podoby, v jaké jej spatřujeme dnes - totiž do funkčně košatého nástroje, s jehož pomocí lze implementovat i velice složité stavové firewally včetně možnosti práce s překladem adres (*Network Address Translation*, dále jen NAT) a "trackingu" spojení (connection tracking). Práce s NAT umožňuje zejména "maškarády" (masquarades), "forwardování" portů (port forwarding) a "přesměrovávání" (redirecting). Rusty Russel se i nadále aktivně účastní vývoje projektu, ovšem aktuální osobou číslo jedna je dnes vynikající Harald Welte.

Několik poznámek úvodem

Napsat kvalitní script, který nastaví slušný firewall není zcela jednoduché, ale zase není zapotřebí z toho dělat až přílišnou vědu. Celková koncepce *Netfilteru* je poměrně složitá, ale věřte mi, že je zároveň až geniálně jednoduchá. Každý nadto k problematice nakonec přistupuje docela jinak, než jak mu radí druzí. Proto prosím přistupujte k tomuto dokumentu spíše nezávazně.

Lidé také uvažují rozdílně, když píší script pro nastavení tabulek *Netfilteru*. Existují v zásadě dva způsoby, jak můžete při psaní firewallu přemýšlet:

- jako člověk píšící nějaký script
- jako člověk, který programuje

V čem se oba přístupy liší? Řekněme, že když si jen tak "scriptujeme", pak v podstatě neděláme nic jiného, než že tabulky paketového filtru plníme nějakými daty pomocí příslušného prográmku (obvykle `/sbin/iptables`), přičemž nás příliš nezajímá, co to vlastně vnitřně dělá.

Avšak "programujeme-li", pak si klidně v pojmosloví Netfilteru můžeme nahradit slovo "řetězec" (chain) slovem "funkce". Pohybujeme se pak ve sféře kdy nám Netfilter/IPtables umožňuje procedurální vyjádření našich síťově-bezpečnostních potřeb. Netfilter je totiž vnitřně prakticky plnohodnotný programovací jazyk, v němž lze krom zmíněných "funkcí" implementovat i podmínky a dokonce cykly.

Oba zmíněné způsoby uvažování mají význam především psychologický. Praktický efekt je nulový a výsledná tabulka v obou případech stejná. Někomu se prostě firewally píší lépe tak a někomu tak. Rusty Russel například volí druhý ("programovací") způsob. Ostatně aby ne! On je přeci vynikající programátor. Avšak i my, kteří třeba programovat ani vůbec neumíme, se nyní směle pokusíme naučit nastavit svůj první firewall.

V této kapitolce jste zaslechli několik důležitých pojmů jako například "chain" a "tabulka". Vysvětlíme si je v následující sekci, která pojednává obecně o koncepci *Netfilteru*.

Koncepce Netfilteru

Paketový filter je software, který prohlíží hlavičky procházejících paketů a rozhoduje co se má s kterým paketem stát. Typicky může být packet *zahozen* (`DROP`), *akceptován* (`ACCEPT`) nebo popř může být informace o něm *zalogována* (`LOG`). Obecně těmto akcím říkáme v terminologii *Netfilteru* "cíl" (target), neboli co se s paketem má stát, kam ho chceme směřovat apod. Lze s ním samo sebou dělat mnohé další, i třeba velice složité, kejkle, pro něž si obvykle vytváříme targety vlastní. Pro začátek však vystačíme s těmito třemi akcemi.

Dnešní svět je nebezpečný na všech úrovních reality, tedy i na úrovni reality počítačových sítí. Chceme proto mít kontrolu nad tím, co se k nám dostane, co od nás odchází. O něčem bychom rádi věděli, něčeho se raději zbavíme bez povšimnutí. V obecné rovině nám přesně toto projekt Netfilter umožnuje. Program `iptables` je součástí projektu *Netfilteru* a slouží k nastavování paketového filtru. Jeho manuálová stránka je velice rozsáhlá a dobře napsaná. Po pochopení základních způsobů konfigurace *Netfilteru* se v ní již pohybujeme poměrně lehce.

Pro začátek máme k dispozici tři vestavěné řetězce, do nichž můžeme ukládat nová pravidla:

- `INPUT`
- `OUTPUT`
- `FORWARD`

Program `iptables` nám umožňuje několik základních akcí. Pohodlně můžeme:

- vytvořit nový řetězec pravidel (chain) - `iptables -N`
- vymazat prázdný řetězec - `iptables -X`
- změnit defaultní politiku pro vestavěné (build-in) retězce `iptables -P`
- vypsat si pravidla z konkretního retězce - `iptables -L`
- vyprázdnit (flush) pravidla z retězce - `iptables -F`
- vynulovat čítače paketů a byl na všech pravidlech řetězce - `iptables -Z`

Dále máme k dispozici několik způsobů jak nakládat s pravidly uvnitř specifikovaného řetězce:

- přidat pravidlo na konec řetězce - `iptables -A`
- vložit pravidlo do řetězce - `iptables -I`
- zaměnit pravidlo jiným pravidlem `iptables -R`
- smazat pravidlo z řetězce - `iptables -D`

Základní možnosti filtrování

Teď, když víme, co všechno můžeme dělat na úrovni celých řetězců pravidel, přejdeme k pravidlům samotným a podíváme se, co všechno nám *Netfilter* umožní specifikovat. Pro přehlednost si uvedeme jen ty nejdůležitější věci:

Zdrojovou a cílovou adresu lze specifikovat pomocí přepínačů `--source` (zdroj) a `--destination` (cíl). Oba přepínače mají i své zkrácené verze `-s` či `--src` a `-d` či `--dst`. Lze specifikovat jak symbolické jméno (např. `localhost` nebo `www.abclinuxu.cz` - tuto variantu nedoporučuji) tak IP adresu či celou síť (např. `127.0.0.1`, `12.34.56.78/32`, `98.76.54.32/27`, pro všechny adresy vůbec pak `0/0`).

Negace většiny specifikací lze docílit znakem vykřičníku. Např. `-s ! 127.0.0.1` vyhoví kterémukoli paketu, který nepřichází z `localhost`. Negaci, nebo chcete-li "inverzi", lze použít pro většinu určujících pravidel, je však vždy raději lepší konzultovat manuálovou stránku `man iptables`.

Protokol lze určit pomocí přepínače `--protocol` (zkráceně `-p`). Příkladem může být `-p ! tcp`, čemuž by vyhověly všechny datagramy, které nejsou TCP.

Důležitá je též možnost určit síťové zařízené, na němž chceme filtrovat. Toho lze docílit přepínačema `--in-interface` (zkráceně `-i`) pro vstupní interface a `--out-interface` (zkráceně `-o`) pro výstupní interface.

Je nutné pamatovat na skutečnost, že řetězec `INPUT` nemá výstupní interface a že řetězec `OUTPUT` nemá vstupní interface. Obě, vstupní i výstupní, zařízení má pouze řetězech `FORWARD`.

Dále chceme-li specifikovat například zařízení `eth1`, `eth2` a `eth3`, lze tak udělat v jediném pravidle pomocí znaku "plus" např. takto: `-i eth+`.

Je-li zapotřebí filtrovat či jinak nakládat s *fragmentovanými pakety*. které vznikají vetšinou díky tomu, že příliš velké pakety některá zařízení neumí přenést, lze tak učinit pomocí přepínače `--fragment` (zkráceně `-f`).

Poté, co jsme probrali nutné minimum k `iptables`, obrátíme pozornost k jejich rozšířením.

Rozšíření `iptables`

... a další kapitoly ... (prosím o rezervaci, -- Matouš)

Ukázkový firewall

Ukázkový script na nastavení firewallu může vypadat např. takto.

```
---CUT---
#!/bin/bash

# eth0 - nas interface do sveta
# lo+ - viz `man iptables'

# branime se proti orfismu-proofingu, "mrtvym" chybovym hlaskam a IP spoofingu
if [ -e /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts ]; then
        /bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
fi

if [ -e /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses ]; then
        /bin/echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
fi

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

# uplne vymazeme stary firewall
/bin/cat /proc/net/ip_tables_names | while read TABLE; do
        /sbin/iptables -t $TABLE -L -n | while read C CHAIN REST; do
                if [ "X$C" = "XChain" ]; then
                        /sbin/iptables -t $TABLE -F $CHAIN
                fi
        done
        /sbin/iptables -t $TABLE -X
done

# a vsechno vynulujeme
/sbin/iptables -Z
/sbin/iptables -t nat -Z
/sbin/iptables -t mangle -Z

# uplne vypnout firewall lze prepinacem "stop"
if [ "$1" == "stop" ]; then
        # defaultni politika bude akceptovat
        /sbin/iptables -P OUTPUT  ACCEPT
        /sbin/iptables -P INPUT   ACCEPT
        /sbin/iptables -P FORWARD ACCEPT
        # a smitec
        exit 0;
fi

# defaultni politika bude zahazovat
/sbin/iptables -P OUTPUT  DROP
```

```
/sbin/iptables -P INPUT   DROP
/sbin/iptables -P FORWARD DROP

# povolime provoz na loopbacku
/sbin/iptables -A OUTPUT -o lo+ -j ACCEPT
/sbin/iptables -A INPUT  -i lo+ -j ACCEPT

# uplne se odriznete od sveta prepinacem "panic"
if [ "$1" == "panic" ]; then exit 0; fi

# povolime vsechna odchozi spojeni
/sbin/iptables -A OUTPUT -p tcp  -o eth0 -j ACCEPT
/sbin/iptables -A OUTPUT -p udp  -o eth0 -j ACCEPT
/sbin/iptables -A OUTPUT -p icmp -o eth0 -j ACCEPT

# spatne navazovana spojeni, fragmenty a nesmysly zahazujeme
/sbin/iptables -A INPUT -i eth0 -m state --state INVALID -j DROP
# /sbin/iptables -A INPUT -i eth0 -f -j DROP
/sbin/iptables -A INPUT -i eth0 -p tcp ! --syn -m state --state NEW -j DROP

# branime se proti SYN floodu
/sbin/iptables -N STOP_FLOODS
/sbin/iptables -A STOP_FLOODS -m limit --limit 1/s --limit-burst 5 -j RETURN
/sbin/iptables -A STOP_FLOODS -j DROP
/sbin/iptables -A INPUT -i eth0 -p tcp --syn -j STOP_FLOODS

# povolime jiz navazana nebo nami iniciovana TCP a UDP spojeni
/sbin/iptables -A INPUT -i eth0 -p tcp  -m state --state ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -A INPUT -i eth0 -p udp  -m state --state ESTABLISHED,RELATED -j ACCEPT

# nektere ICMP pakety preci jenom povolime, ale floodovat nas nebudou
/sbin/iptables -N CHOOSE_ICMP
/sbin/iptables -A CHOOSE_ICMP -p icmp -m state --state ESTABLISHED,RELATED    -j ACCEPT
/sbin/iptables -A CHOOSE_ICMP -p icmp --icmp-type 0 -m length --length 28:84  -j ACCEPT
/sbin/iptables -A CHOOSE_ICMP -p icmp --icmp-type 3 -m length --length 28:84  -j ACCEPT
/sbin/iptables -A CHOOSE_ICMP -p icmp --icmp-type 8 -m length --length 28:84 \
                                      -m limit --limit 1/s --limit-burst 5 -j ACCEPT
/sbin/iptables -A CHOOSE_ICMP -p icmp --icmp-type 11 -m length --length 28:84 -j ACCEPT
/sbin/iptables -A INPUT -i eth0 -j CHOOSE_ICMP

# povolime vzdalene pripojeni pomoci SSH
/sbin/iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT

# povolime vzdalene pripojeni pomoci SSH z IP.AD.RE.SA pouze
# /sbin/iptables -A INPUT -i eth0 -s IP.AD.RE.SA -p tcp --dport 22 -j ACCEPT

# odmitneme spojeni na AUTH a resetujeme ho
/sbin/iptables -A INPUT -i eth0 -p tcp --dport 113 -j REJECT --reject-with tcp-reset

# konec
---CUT---
```

Případný port-scan programem *NMap* pak dopadne asi takto:

```
---CUT---
#> nmap -P0 -TInsane NA.SE.IP.ADRESA

Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2005-12-24 16:48 CET
All 1663 scanned ports on SYMBOLICKE.JMENO.TLD (NA.SE.IP.ADRESA) are: filtered

Nmap finished: 1 IP address (1 host up) scanned in 87.792 seconds
---CUT---
```

Zkusme si stroj "pingnout" paketem velkým 1400 bytů.

```
---CUT---
#> ping -c1 -s 1400 NA.SE.IP.ADRESA
PING NA.SE.IP.ADRESA (NA.SE.IP.ADRESA) 1400(1428) bytes of data.

--- NA.SE.IP.ADRESA ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
---CUT---
```

Normální `ping` ale projde.

```
---CUT---
$> ping -c1 NA.SE.IP.ADRESA
PING NA.SE.IP.ADRESA (NA.SE.IP.ADRESA) 56(84) bytes of data.
64 bytes from NA.SE.IP.ADRESA: icmp_seq=1 ttl=58 time=12.6 ms

--- NA.SE.IP.ADRESA ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 12.632/12.632/12.632/0.000 ms
---CUT---
```

Stejně tak projde `mtr` bez změny parametrů (používá ICMP pakety, které se v defaultním nastavení vejdou do specifikovaného rozsahu, mají velikost 64 bytů). Zkusíme-li ale `mtr -i 0.1 NA.SE.IP.ADRESA`, tak už náš stroj bude vykazovat ztrátovost paketů.

Přehled příkazů

Základní příkazy

Zde je seznam základních příkazů a jejich stručný popis. Více k příkazu zjistíte vždy z jeho manuálové stránky (`man prikaz`).

Pokud používáte Konqueror (součást KDE), můzete napsat do řádku adresy "man:prikaz" - tento postup je velice výhodný, manuál je hezky zformátován, lehce tisknutelný a také vyhledávání v něm je snadné.

V případě nutnosti ukončení programu se používá [Ctrl]+[C], a pokud by šlo opravdu do tuhého, pak použijte [Ctrl]+[D]

alsaconf -Nastavení ALSA - Advancend Linux Sound Architecture.

apt-get -Balíčkovací systém distribucí s .deb balíčky (Debian, Ubuntu).

at -Spouští příkazy v zadanou dobu.

cat -Vypíše nebo spojí soubory.

cd -Přechod mezi adresáři. Pokud nezadáte žádný parametr, přejde se do ~.

clear -Vyčistí okno terminálu.

cp -Kopírování souborů.

df -Volné místo na oddílech.

du -Zobrazí součet velikosti souborů v adresáři.

date -Vypíše datum a čas.

eject -Vysunutí a zasunutí mechaniky.

file -Zobrazí typ souboru.

fmt -Provádí jednoduché formátování textu.

free -Obsazení paměti.

grep -Vyhledání řetězce.

gzip -Zabalí a rozbalí - (de)komprimuje - data.

halt -Vypne systém.

chmod -Mění přístupová práva k souboru.

chown -Mění vlastníka souborů / adresářů.

iconv -Konvertuje znakovou sadu.

init -Spustí nebo ukončí aplikce podle nstavení v /etc/init.d

kill -Ukončuje proces dle zadaného PID.

killall -Ukončuje všechny procesy se zadaným jménem.

less -Vypisuje obsah textových souborů.

ln -Vytváří odkazy na soubory.

locate -Slouží ke hledání souborů. Před použitím je nutno spustit updatedb.

logout -Odhlásí vás ze systému.

ls -Vypíše obsah aktuálního adresáře.

mkdir -Vytváří adresáře.

mount -Připojuje disky.

more -Podobně jako less umožňuje pozastavit výpis.

most -Vylepšená verze programů less a more.

mv -Přesouvá nebo přejmenovává soubory.

ping -Provede měření odezvy zadaného počítače nebo jiného síťového prvku.

passwd -Mění heslo uživatele.

pwd -Vypíše cestu k aktuálnímu adresáři.

reboot -Restart systému.

reset -Resetování terminálu.

rm -Maže soubory nebo adresáře.

shutdown -Vypne nebo restartuje počítač.

startx -Spustí systém X Window.

su -Přepíná uživatele.

tar -Zabalí a rozbalí - (de)komprimuje - data.

tree -Vypíše obsah adresáře jako strom.

uptime -Zobrazí aktualní čas (hh:mm:ss), dobu jakou je systém spuštěn a dobu která byla potřeba na zpracování této úlohy.

urpm -Balíčkovací systém distribucí používajících RPM (Redhat, Mandriva, Fedora ...).

uname -Zobrazí název jádra, s parametrem -a přidá i verzi, název počítače a systému.

umount -Odpojuje disky.

who -Seznam přihlášených uživatelů.

which -Zobrazí plnou cestu k programu (příkazu shellu).

wine -Spustí binární soubor určený pro systém Mirosoft Windows.

vmstat -Zobrazí statistiku o zaplnění virtuální paměti, využití cpu, přerušení,...

Na co se často ptáme: X Window System - I

XFree 4.3.0

X Window System (dále v článku také XFree nebo X) všichni známe, nebudu se věnovat jeho historii, technologii nebo budoucnosti. Budu se věnovat té nejobyčejnější věci, která pro mnohé uživatele představuje noční můru. Jedná se o nastavení tohoto komplexního systému. Protože má mnoho vlastností (features), jedná se o práci složitou. Lze použít pomocné konfigurační nástroje, které bývají dodávány jednak se samotným XFree, jednak s konkrétní distribucí. Takové řešení bývá obvykle funkční a pro normální provoz dostačující. Někdy ale nastane situace, kdy tyto programy nepomohou. Jedná se o případy, kdy máte příliš starý nebo nový hardware (grafická karta, monitor, displej), nebo ve vaší distribuci daný konfigurační nástroj není nebo nefunguje tak, jak má. Nebo, do třetice, program vám grafický systém nastaví, ale jaksi "málo". Často je použit standardní ovladač nebo parametr, který výkon vašeho hardwaru degraduje (typicky nízká obnovovací frekvence monitoru).

V textu se pokusím vyčerpávajícím způsobem popsat jednotlivé sekce a parametry konfiguračního souboru `/etc/X11/XF86Config`, kterým je nastaveno chování XFree verze 4.3.0. Rád bych upozornil, že se nejedná o poslední finální verzi, protože se ale u aktuální verze 4.4 změnila licence, mnozí producenti distribucí zůstali u verze 4.3, případně přecházejí ke konkurenčním grafickým systémům (x.org, Y Window System).

Rád bych také upozornil, že syntaxe a dělení jednotlivých částí konfiguračního souboru se od starších verzí změnila a tento materiál se na ně nevztahuje. Veškeré informace, a to i ke starším verzím, lze nalézt na stránkách XFree.

Před započetím práce si ověřte, že soubor, který zpracováváte, je při spuštění XFree opravdu interpretován. Pracujete-li jako root, což je pro ladění běhu XFree nutné a máte-li v domovském adresáři soubor `XF86Config`, což se může stát, je interpretován právě on, nikoliv soubor ze standardní cesty `/etc/X11/`.

Také si okamžitě vytvořte nejméně jednu zálohu. To je ostatně vhodné opakovat pokaždé, když se vám bude zdát, že máte část souboru odladěnou. Soubor s veškerými výpisy (log) se nachází v adresáři `/var/log/` a nejprve se jmenuje `XFree.0.log`, starší verze mají ještě navíc koncovku `.old`, případně se vnitřní nula mění na čtyřku.

Struktura konfiguračního souboru

Konfigurační soubor se skládá z několika sekcí, na jejichž pořadí obvykle (viz níže) nezáleží. Každá ze sekcí má tvar:

```
Section "Jmeno_sekce"
  Polozka_sekce
  Polozka_sekce
  Polozka_sekce
EndSection
```

Místo pseudonázvu `Jmeno_sekce` se doplňuje některý z názvů konkrétních sekcí:

Název sekce	Význam a použití
`ServerLayout`	Integrující konfigurace serveru
`Files`	Cesty k souborům písem, palety apod.
`ServerFlags`	Příznaky X-serveru
`Module`	Načítání modulů XFree
`InputDevice`	Definice vstupního zařízení
`Device`	Definice grafického výstupního zařízení
`VideoAdaptor`	Definice Xv grafického adaptéru
`Monitor`	Definice monitoru
`Modes`	Definice grafických režimů
`Screen`	Konfigurace obrazovky
`DRI`	Konfigurace DRI (Direct Rendering)
`Vendor`	Konfigurace závislá na výrobci

Syntaxe konfiguračního souboru

Při interpretaci klíčových slov nehraje roli velikost písmen. Znak "_" (podtržítko) je ignorován; u některých parametrů jsou ignorovány také znaky označující volný prostor (znaky pro mezeru a tabelátor). Každá položka v souboru zabírá obvykle jeden řádek, který se skládá z klíčového slova obvykle následovaného jedním nebo více argumenty. Tyto argumenty mohou nabývat hodnot celých čísel, reálných čísel nebo textových řetězců:

Název	Význam, hodnoty, rozmezí
`integer`	Celé číslo v desítkové, šestnáctkové (musí začínat "0x") nebo osmičkové (musí začínat "0") soustavě.
`real`	Reálné číslo s plovoucí čárkou.
`string`	Textový řetězec uzavřený do uvozovek.

Existuje speciální klíčové slovo `Option` (doslova "volba, možnost"), kterým lze serveru předat nestandardní parametr s volnou strukturou. `Option` má jeden, nebo dva argumenty. První je název volby a možný druhý je parametr volby. Ten může kromě výše uvedených typů (integer, real, string) nabývat také hodnot

Název	Význam, hodnoty, rozmezí
`boolean`	Stavový příznak, který nabývá pouze dvou hodnot: platí, nebo neplatí. Pokud není uvedena hodnota, předpokládá se hodnota "platí". Tu lze také vyjádřit následujícími znaky, které se považují za rovnocenné: 1, on, true, yes. Hodnota "neplatí" se vyjadřuje analogicky: 0, off, false, no.
`frequency`	Hodnota frekvence se vyjadřuje reálným číslem, které může být doplněno jednotkami Hz, k, kHz, M, MHz. Pokud označení jednotky chybí, je odhadnuto z rozmezí, které by mělo být schopno použít dané zařízení. Pochopitelně je vhodné jednotku vždy uvést. Nehrozí ani tak zničení zařízení (monitoru), jako spíše zavlečení chyby.

Všechny parametry volby `Option`, nejenom řetězce, musejí být uzavřeny v uvozovkách. Pokud parametr začíná předponou "No", je jeho hodnota "neplatí". Následující příklady vyjadřují naprosto totéž (zakázání akcelerace):

```
Option "Accel" "Off"
Option "NoAccel"
Option "NoAccel" "On"
Option "Accel" "false"
Option "Accel" "no"
```

Popis jednotlivých sekcí

ServerLayout

Tato sekce má v hierarchii nejvyšší váhu. Svazuje totiž konkrétní vstupní a výstupní zařízení. Vstupní zařízení jsou popsána v sekcích `InputDevice`. Parametr začínající `Core-` určuje standardní vstupní zařízení. V systému musí být přesně jedno pro jednotlivé kategorie (myš, klávesnice); není-li explicitně označeno, je použito první, které lze technicky využít. Toto lze určit také v patřičné sekci `InputDevice` nebo na příkazovém řádku.

Výstupní zařízení se obvykle skládají z vícero nezávislých komponent (grafické karty a monitoru). Tato zařízení jsou svázána v sekcích `Screen`; na tyto sekce se pak odvolává v sekci `ServerLayout`. Grafické karty jsou definovány v sekcích `Device` a monitory v sekcích `Monitor`. Záměrně vše uvádím v množném čísle, protože XFree podporuje více zařízení v jednom sezení (např. dvě karty a dva monitory, nebo jedna karta se dvěma výstupy na dva monitory apod.). Téma na samostatný článek.

```
Section "ServerLayout"
  Identifier "XFree86 Configured"
  Screen 0 "Screen0" 0 0
  InputDevice "Mouse0" "CorePointer"
  InputDevice "Keyboard0" "CoreKeyboard"
EndSection
```

Files

```
Section "Files"
  RgbPath "/usr/X11R6/lib/X11/rgb"
  ModulePath "/usr/X11R6/lib/modules"
  FontPath "/usr/X11R6/lib/X11/fonts/TTF/"
  FontPath "/usr/X11R6/lib/X11/fonts/misc/"
  FontPath "/usr/X11R6/lib/X11/fonts/Speedo/"
  FontPath "/usr/X11R6/lib/X11/fonts/Type1/"
  FontPath "/usr/X11R6/lib/X11/fonts/75dpi/"
  FontPath "/usr/X11R6/lib/X11/fonts/100dpi/"
  #FontPath "unix/:7100"
  FontPath "tcp/192.168.1.10:7100"
EndSection
```

Tato sekce je určena k předání informací o souborech, které jsou nutné pro běh serveru. Některé z těchto informací lze předávat serveru jako parametr při spuštění, nebo za běhu, např. programem `xset`. Takové parametry mají větší váhu než ty, které jsou uvedené v konfiguračním souboru. Položky této sekce jsou následující.

Parametr	Hodnota	Význam
`FontPath`	`adresarova_cesta` nebo `protokol/klient:cislo_portu`	Adresářová cesta k souborům s fonty. Je možné (a obvyklé) uvádět více těchto parametrů. Obvyklá cesta k souborům s fonty je `/usr/X11R6/lib/X11/fonts/`. Pokud daný adresář neexistuje nebo neobsahuje žádné soubory písem nebo není vytvořen jejich seznam (soubory `fonts.dir`, `fonts.alias` a `fonts.scale`), XFree oznámí, že odstraňuje tuto položku ze seznamu. Obvyklý případ hlavně při (nesprávném) použití TrueType fontů. Druhá hodnota je případ spolupráce s fontserverem běžícím v síti. Problematika přesahuje tento článek, viz např. článek o fontech v Mozille.
`FontPath`	`adresarova_cesta`	Specifikace souboru s databází barev. K názvu je přidána přípona `.txt`. Soubor obsahuje RGB kódy a názvy barev. Obvykle `/usr/X11R6/lib/X11/rgb`.
`ModulePath`	`adresarova_cesta`	Specifikace adresáře, ve kterém bude systém XFree hledat moduly. Ty se obvykle používají pro akceleraci, podporu TrueType a Type1 písem, zařízení Video4Linux apod. Může být použito opakovaně. Obvykle `/usr/X11R6/lib/modules/`.

ServerFlags

```
# Nastavení "dvouhlavé" karty pro dva monitory
Section "ServerFlags"
  DefaultServerLayout "Multihead"
  Option "BlankTime" "0"
  Option "Xinerama" "true"
EndSection
```

Tato sekce je určena k specifikaci globálních vlastností X serveru. Všechny položky v této sekci jsou typu `Option` (viz výše). Jsou interpretovány také některé položky ze starších verzí XFree; těmi se zde ovšem nezabýváme.

Parametr	**Typ argumentu**	**Význam hodnoty**
`DefaultServerLayout`	`string`	Sekce `ServerLayout`, která bude použita jako standardní, pokud není její jméno definováno na příkazové řádce parametrem `-layout`. Je-li nalezeno více sekcí `ServerLayout`, platí **poslední uvedená!**
`NoTrapSignals`	`boolean`	Určeno pro ladění. Rozhodnutí, má-li X server překonat chyby, nebo se naopak ukončit a uložit ladicí informace.
`DontVTSwitch`	`boolean`	Je-li `true`, není možné přepínat se na další terminály pomocí klávesových zkratek Ctrl+Alt+Fn.
`DontZap`	`boolean`	Je-li `true`, není možné ukončit (nebo pouze restartovat) XFree pomocí Ctrl+Alt+Backspace.
`DontZoom`	`boolean`	Je-li `true`, není možné přepínat videorežimy pomocí Ctrl+Alt+šedé Plus a Ctrl+Alt+šedé Minus.
`AllowMouseOpenFail`	`boolean`	Je-li `true`, server se spustí, i když není připojena nebo správně nastavena myš.
`XkbDisable`	`boolean`	Povoluje, nebo zakazuje rozšíření XKEYBOARD, které slouží k přepínání klávesových map. Standardně povoleno.
`BlankTime`	`integer`	Čas (v minutách) definující dobu, než se spustí spořič obrazovky. Lze změnit za běhu pomocí `xset`. Různá prostředí (KDE) tuto hodnotu modifikují.
`StandbyTime`	`integer`	Čas (v minutách) definující dobu, než se monitor přepne do "stand-by" režimu. Lze změnit za běhu pomocí `xset`. Nemusí fungovat u všech videodriverů a monitorů. Nastaveno pouze v případě, že je v sekci `Monitor` definován parametr `DPMS` (viz níže).
`SuspendTime`	`integer`	Čas (v minutách) definující dobu, než se monitor přepne do uspávacího režimu. Lze změnit za běhu pomocí `xset`. Nemusí fungovat u všech videodriverů a monitorů. Nastaveno pouze v případě, že je v sekci `Monitor` definován parametr `DPMS` (viz níže).
`OffTime`	`integer`	Čas (v minutách) definující dobu, než se monitor vypne. Lze změnit za běhu pomocí `xset`. Nemusí fungovat u všech videodriverů a monitorů. Nastaveno pouze v případě, že je v sekci `Monitor` definován parametr `DPMS` (viz níže). Všechny tyto čtyři sekce jsou závislé na hardwaru a jejich chování lze vyzkoušet snad jedině prakticky.

Xinerama	boolean	Zakáže, nebo povolí rozšíření XINERAMA, které umožňuje práci s více monitory a kartami, případně vícehlavými kartami (např. ATI).

Existují další parametry, které mají speciální význam a které se obvykle, tj. ve standardní uživatelské konfiguraci, nepoužívají.

Module

V této sekci se specifikují moduly, které se mají při sezení načíst. Jedná se obvykle o rozšíření X serveru nebo moduly pro práci s různými typy písem (TrueType, Type1, Speedo apod.). Většina ostatních modulů je načítána na žádost, takže se o ně nemusíme starat.

Existují dva způsoby jak specifikovat moduly. Nejčastější a nejpohodlnější je způsob `Load "nazev-modulu"`. Argument `nazev-modulu` je název modulu, nikoliv souboru. Řetězec rozlišuje velikost znaků a nesmí obsahovat předponu `lib` stejně jako přípony `.a`, `.o` nebo `.so`.

Druhý, méně častý, způsob je položka `SubSection` s argumentem názvu modulu a parametry typu `Option`, které jsou předány modulu při jeho načtení. Tento způsob je určen pro opravdu speciální případy, kdy je nutné zcela přesně nastavit parametry hardwaru. Je tedy nutné znát parametry jednotlivých modulů, což jejich autor (v takovém případě spíše výrobce) určitě sděluje v dokumentaci k modulu.

Soubory s moduly jsou hledány v adresářích, které jsme určili v sekci `Files`. Každý z nich může obsahovat podadresáře (např. drivers, input, extensions, fonts), které jsou také prohledávány. Standardní adresář je `/usr/X11R6/lib/modules/`. Jako absolutní minimum se doporučuje nahrát alespoň modul `extmod`, `bitmap` pro správu systémových písem se nahrává automaticky.

```
Section "Module"
  Load "record"
  Load "xaa"
  Load "extmod"
  Load "drm"
  Load "dbe"
  Load "dri"
  Load "v4l"
  Load "GLcore"
  Load "glx"
  Load "xtrap"
  Load "type1"
  Load "freetype"
  Load "xtt"
  Load "speedo"
EndSection
```

InputDevice

Konfigurační soubor může obsahovat (a obvykle také obsahuje) více těchto sekcí. Minimálně dvě jsou vždy přítomny: jedna pro myš a druhá pro klávesnici.

```
Section "InputDevice"
  Identifier "Keyboard0"
  Driver "Keyboard"
  Option "CoreKeyboard"
  Option "MapName" "Standard Keyboard [ pc105 + toggle ]"
  Option "Protocol" "Standard"
  Option "XkbLayout" "cz,us"
  Option "XkbModel" "pc105"
  Option "XkbOptions" "grp:shift_toggle"
  Option "XkbRules" "xfree86"
EndSection

Section "InputDevice"
  Identifier "Mouse0"
  Driver "mouse"
  Option "CorePointer"
  Option "Protocol" "auto"
  Option "Device" "/dev/mouse"
  Option "Buttons" "3"
  #Option "Resolution" "N" #ukázka
  Option "ZAxisMapping" "4 5"
EndSection
```

Celá sekce je platná, je-li na ni odkazováno v sekci `ServerLayout` nebo parametrem z příkazové řádky.

Parametr	Význam hodnoty
`Identifier`	Představuje jedinečné jméno vstupního zařízení.
`Driver`	Typ zařízení, nejčastěji `keyboard` a `mouse`.
`CoreKeyboard`	Definuje hlavní klávesnici, viz také výše.
`CorePointer`	Definuje hlavní polohovací zařízení, viz také výše.
`MapName`	Popis typu klávesnice.
`Protocol`	V případě klávesnice obvykle `standard`. V případě myší dnes již nejlépe `auto`, ale různé typy jsou stále rozlišovány.
`Device`	Platí pro myš, obvykle `/dev/mouse`, což bývá symbolický odkaz např. na `/dev/psaux`.
`Buttons`	X server neumí zjistit počet tlačítek myši (což mě udivuje, ale tak se to píše v dokumentaci). Přednastavená hodnota je `3` tlačítka.
`Resolution`	Rozlišení pro pohyb kurzoru myši.
`ZAxisMapping`	Namapování ostatních tlačítek myši. V uvedeném případě je pomyslné tlačítko 4 (ve skutečnosti pohyb kolečkem od sebe) a 5 (pohyb kolečkem k sobě) namapováno na osu Z, tedy posunování obsahu okna (scrolling).

`XkbRules`	Soubory pravidel pro mapování klávesnice, soubory se nacházejí v adresáři `/usr/X11R6/lib/X11/xkb/rules/`.
`XkbModel`	Název modelu klávesnice, její typ.
`XkbLayout`	Rozložení klávesnice.
`XkbVariant`	Varianty rozložení.
`XkbOptions`	Další vlastnosti a konfigurace ovládání klávesnice.

Další parametry jsou pro speciální účely, případně nedokumentované. Konfigurace klávesnice prošla během vývoje XFree mnoha změnami a zejména komplexní prostředí jako KDE tento vývoj výrazně ovlivnila. Už fakt, že návody k nastavení klávesnice jsou nejednotné, stejně jako technologie, svědčí o tom, že řešení není ideální. Více informací lze nalézt v již zmiňovaném dokumentu.

Příště se budeme věnovat nastavení výstupních zařízení.

Na co se často ptáme: X Window System - II

V dnešním dílu se zaměříme na výstupní zařízení a jejich nastavení.

Device

Konfigurační soubor může obsahovat těchto sekcí několik. Přítomná musí být alespoň jedna. V této sekci konfigurujeme grafickou kartu. V případě vícehlavých karet je nutné vytvořit dva stejné záznamy; pokud se však každá hlava identifikuje jiným `BusID` číslem, pracujeme jakoby se dvěma oddělenými kartami.

Parametr	**Význam hodnoty**
`Identifier`	Představuje jedinečné jméno výstupního zařízení.
`Driver`	Ovladač grafické karty.
`BusID`	Specifikuje umístění grafické karty ve smyslu `PCI:sběrnice:zařízení:funkce`. Např. `PCI:1:0:0` je ale AGP karta. Tuto hodnotu lze zjistit spuštěním X serveru s parametrem `-scanpci`, nebo příkazem `lspci`. `01:00.0 VGA compatible controller: ATI Technologies Inc Radeon RV100 QY [Radeon 7000/VE]`
`Screen`	Obvykle nemá význam v případě jedné karty. V případě více karet nebo vícehlavých karet určuje tento parametr, kterou obrazovku hlava/karta obsluhuje. Pojem "obrazovka" chápejme ve smyslu "stínítko", tedy plocha k zobrazení. Je úzce svázána se zařízením monitoru (sekce `Monitor`). Čísluje se od 0 do počet obrazovek-1.
`Chipset`	Doporučuje se nenastavovat, pokud to není nezbytně nutné. Moduly jsou schopné samy přítomný čipset rozpoznat. Osvědčí se ale u starých karet, které mají podobné, ale přesto se mírně lišící, čipsety (např. S3).
`RAMDAC`	Význam tohoto čipu objasní spíše elektrotechnik. Moduly opět samy poznají typ RAMDAC a chovají se podle toho.

`Clocks`	Frekvence časovače v MHz. Dnes pro moderní karty nevyužitelné; snad jen pro ty staré.
`VideoRAM`	Množství operační paměti karty v kilobajtech. Dnes již obvykle nevyužívané pro karty v pravém slova smyslu (zasunují se do sběrnice), naopak praktické pro integrované grafiky, např. SiS.

Existuje ještě několik parametrů, s jejichž pomocí je snad možné dosáhnout určitých výsledků. Neslyšel jsem ale o tom, že by se dal takto změnit (zvýšit) výkon karty.

Kromě toho má každý čipset své specifické volby, které jsou velmi důležité, protože výkon zásadně ovlivňují. Následující čipsety mají své manuálové stránky, kde lze konkrétní volby nalézt: chips, cirrus, cyrix, fbdev, glide, glint, i128, i740, i810, imstt, mga, neomagic, nv, radeon, r128, rendition, savage, s3virge, siliconmotion, sis, sunbw2, suncg14, suncg3, suncg6, sunffb, sunleo, suntcx, tdfx, tga, trident, tseng, v4l, vesa, vga.

Problémy s firemními ovladači mnozí z vás znají, proto znovu upozorňuji, že tyto manuálové stránky pojednávají o modulech XFree, nikoliv jednotlivých výrobců, kteří dodávají vlastní dokumentaci. Jako příklad uvádím konfiguraci obstarožního Radeonu 7000 (podrobnosti jsou jako obvykle v manuálové stránce).

```
Section "Device"
  Identifier "radeon0"
  Driver "ati"
  #Driver "radeon" # vyjde nastejno
  VendorName "ATI Radeon VE"
  Option "AGPMode" "4"
  Option "ForcePCIMode" "on"
  Option "AGPFastWrite" "on"
  #Option "EnablePageFlip" "on"
  #Option "composite_sync" "off"
  BusID "PCI:1:0:0"
  Screen 0
EndSection
```

VideoAdaptor

O této famózní sekci jsem se všude dočetl jediné: Nikdo neví, kdy a proč vznikla, jakou má funkci a proč vlastně existuje.

Monitor

V této sekci se definují monitory - fyzická zařízení, která budou později svázána s obrazovkou. Monitor je zobrazovací zařízení, které má za úkol zobrazit přijatý signál v co nejlepší kvalitě. Tím máme zejména na mysli:

1. přijatelné rozlišení;
2. bitovou hloubku;
3. frekvenci.

Platí přitom pravidlo, že všechny podmínky musejí být splněny zaráz; teprve poté je cíle dosaženo.

Přijatelné pracovní rozlišení se pohybuje od hodnot 800*600 (dávná minulost), 1024*768 nebo

1152*864(nedávná minulost), 1280*1024 (současnost) až po 1200*1600 a vyšší (budoucnost). Mám pochopitelně na mysli moderní stroje, protože všechna rozlišení v tomto rozsahu jsou dnes podle okolností používaná.

Bitová hloubka čili schopnost zobrazit určitou paletu barev je zásadní pro dnešní mediální svět, a proto cokoliv pod 16bitů nemá smysl (vyšší už je pouze hloubka 32bitová, takže možností tolik není).

Frekvence je zase nejdůležitější pro zdraví. Člověk, který má špatně nastavený monitor, hazarduje se zdravím - a nejedná se jen o zrak. Byť není správcem svého počítače (v zaměstnání), měl by požadovat maximální hodnoty, které hardware umožňuje. Podle mého laického názoru lze takto aplikovat normy Bezpečnosti a ochrany zdraví při práci (BOZP), které jsou součástí/dodatkem Zákoníku práce.

Jak se pozná špatně nastavená frekvence monitoru? Necvičené oko to na první pohled nijak nepozná. Až se "ozve" krční páteř, hlava, záda a kříž, poznáte. Cvičené oko se zadívá těsně vedle monitoru a periferním viděním se snaží poznat, jestli se určitá místa obrazu tetelí a chvějí. Pokud ano, je nastavena frekvence kolem 60Hz (někdy i trochu méně!). Na moderních (mladších pěti let) monitorech stačí stisknout tlačítko OSD (On Screen Display), což je informační nabídka monitoru. Slouží k ovládání a kalibraci monitoru a obvykle obsahuje základní informace o rozlišení a frekvenci. Správnou frekvenci však musíme nastavit softwarově.

Parametr	**Význam hodnoty**
`VendorName`	Výrobce monitoru. Jakýkoliv řetězec.
`ModelName`	Model monitoru. Jakýkoliv řetězec.
`HorizSync`	Horizontální obnovovací frekvence monitoru. Pokud máte manuál, bývá obvykle v tabulce kolem strany osm až dvanáct. Zadává se seznam hodnot oddělená čárkami, nebo rozmezí. Např. `40-120`.
`VertRefresh`	Vertikální obnovovací frekvence. Zadává se stejne jako `HorizSync`, pouze hodnoty jsou nominálně nižší, např. `40-80`.
`DisplaySize vyska sirka`	Rozměr fyzické plochy monitoru v milimetrech. Použito pro vypočítání rozlišení pro obrazovku. Opravdu je jediná možnost vzít pravítko a měřit.
`Gamma`	Gamma korekce monitoru. Buď jediná hodnota, nebo tři složky RGB v rozmezí 0-1.
`Mode`	Volba pro vytvoření nového režimu; vhodnější řešení je `Modeline`
`Modeline "jmeno" definice`	V podstatě instrukce pro vytvoření nového režimu monitoru. Předem varuji, že hodnoty si nelze vymyslet a že jejich nesprávné užití může monitor poškodit. Taky je ale pravda, že moderní monitory se umí bránit a prostě zobrazí hlášení "Signal out of range" (Signál mimo rozmezí) nebo podobné. O modelines více níže.
`DPMS`	Lze zapnout, nebo vypnout úsporný režim monitoru.
`DDCmode`	Zakáže, nebo povolí příjem informací o monitoru, tj. o frekvenčním rozmezí, předdefinovaných režimech apod.

Moderní grafické karty jsou schopné signál vysílat na vysokých frekvencích, pro monitory má význam rozmezí (75)85 až 120 Hz. Pro LCD displeje je pojem "frekvence" irelevantní - zobrazovací mechanismus je odlišný od toho, na kterém pracuje monitor.

Pokud tedy zjistíte, že rozlišení, které vidíte, je realizováno na frekvenci 60 Hz, a přitom máte nastavený správný modul pro grafickou kartu, zkuste změnit (roztáhnout) frekvenční rozsah monitoru. Monitor by si měl najít "tu svou". Je také vhodné experimentovat (ne vždy je to přínos) s parametrem `DDCmode`. Téměř každý monitor má předdefinované režimy. Je na vás, abyste ho donutili je použít. Některý to udělá automaticky, jinému je nezbytné vnutit přesné frekvenční rozmezí (metodou pokus-omyl) za pomoci `DDCmode`.

*Příklad z praxe. Starší digitální monitor Fujitsu ErgoPro x152 z bazaru (tehdy cca 1500 Kč) reaguje na téměř jakýkoliv požadavek tím, že zvolí nejvyšší frekvenci, tj. od 75 do 85 Hz. Oproti tomu nový Hyundai ImageQuest QV790, údajně nejmenší 17" monitor na světě, místo aby zvýšil frekvenci, zvýší rozlišení i za cenu vyloženě nízké frekvence 60Hz. Stále jsem místo požadovaných 1280*1024*16bit*85+ dostával 1600*1200*16bit*60-. Sladění karty a monitoru trvalo několik hodin.*

Pojďme k modelines. O překlad se ani nesnažím, snad "definice režimu". Jedná se o řádek, ve kterém zadáme fyzické hodnoty pro nový režim. Počet režimů je u každého monitoru jinak omezený (i když by měly platit standardy) a záleží na monitoru, zda je schopen hodnoty použít. Existují tři způsoby jak řádek s `Modeline` získat.

1. V souboru `/var/log/XFree.0.log` se zobrazují řádky `Modeline`, které jsou k dispozici, ale nebyly použity.

   ```
   (II) RADEON(0): Not using mode "1152x864" (width too large \\
      for virtual size)
   (II) RADEON(0): Not using default mode "1152x768" (width too \\
      large for virtual size)
   (--) RADEON(0): Virtual size is 1024x768 (pitch 1024)
   (**) RADEON(0): *Mode "1024x768": 94.5 MHz, 68.7 kHz, 85.0 Hz
   (II) RADEON(0): Modeline "1024x768"   94.50  1024 1072 1168 \\
      1376  768 769 772 808 +hsync +vsync
   (**) RADEON(0):  Default mode "832x624": 57.3 MHz, 49.7 kHz, \\
      74.6 Hz
   ```

   Stačí vyseknout požadovanou část a umístit ji do sekce `Monitor` v konfiguračním souboru.
2. Online generátory. Ne vždy fungují na 100 %, ale obvykle dostačují. Např. XTiming, Colas XFree Modeline Generator nebo Modeline Tool.
3. Ruční vytvoření za pomoci programu xvidtune nebo videogen. Vhodné pouze pro adrenalinové sportovce; já jsem si vždycky pomocí xvidtune spolehlivě přizabil systém.

Ukázka sekce `Monitor`.

```
Section "Monitor"
  Identifier "Monitor0"
  VendorName "Monitor Vendor"
  ModelName "Monitor Model"
  HorizSync 40-100
  VertRefresh 60-130
  Option "DDCmode" "off"
  Option "dpms" "off"
Modeline "1600x1200" 202.50 1600 1664 1856 2160 1200 1201 1204
1250 +hsync +vsync
Modeline "1400x1050" 155.80 1400 1464 1784 1912 1050 1052 1064
1090 +hsync +vsync
Modeline "1280x1024" 157.50 1280 1344 1504 1728 1024 1025 1028
1072 +hsync +vsync
```

```
Modeline "1280x1024" 135.00 1280 1296 1440 1688 1024 1025 1028
1066 +hsync +vsync
Modeline "1152x864" 108.00 1152 1216 1344 1600 864 865 868 900
+hsync +vsync
Modeline "1024x768" 94.50 1024 1072 1168 1376 768 769 772 808
+hsync +vsync
Modeline "1280x960" 148.50 1280 1344 1504 1728 960 961 964 1011
+hsync +vsync
Modeline "1280x960" 108.00 1280 1376 1488 1800 960 961 964 1000
+hsync +vsync
EndSection
```

Modes

Grafické režimy zde lze definovat i nezávisle na zobrazovacím zařízení. Nemám s tím zkušenosti.

Screen

Soubor může opět obsahovat více těchto sekcí. V každé z nich se spojuje zařízení grafické karty (sekce `Device`) se zobrazovacím zařízením (sekce `Monitor`), čímž vznikne obrazovka (screen). Každé originální spojení má své číslo. (V případě více karet je již uvedeno v sekci `Device`).

```
Section "Screen"
  Identifier "Screen0"
  Device "radeon0"
  Monitor "Monitor0"
  DefaultDepth 16
  Subsection "Display"
    Depth 16
    #Modes "1280x1024" "1152x864"
    Modes "1024x768"
  EndSubsection
EndSection
```

Každá sekce `Screen` musí obsahovat podsekci `Display`. V ní určujeme buď standardní, nebo dříve - pomocí `Modeline` - vytvořené režimy. Myslím, že ukázka říká vše a je dostatečně srozumitelná. Existuje také množství velmi specializovaných parametrů, které povolují, či zakazují, různá rozšíření X serveru. Nikdy jsem je nepotřeboval a ani neznám jejich účel.

DRI

Nastavení Direct Rendering Interface, což je rozhraní pro hry, není témeř žádné. Je vlastně pouze nutné stanovit práva pro zařízení tak, aby ho mohli využívat všichni uživatelé. Podmínkou je samozřejmě načtený odpovídající modul a podpora v jádře, ale to je jiná kapitola.

```
Section "DRI"
  Mode 0666
EndSection
```

Příště se zaměřím na konkrétní dotazy z diskuzí.

Na co se často ptáme: X Window System - III

Uživatelské dotazy

Modelines

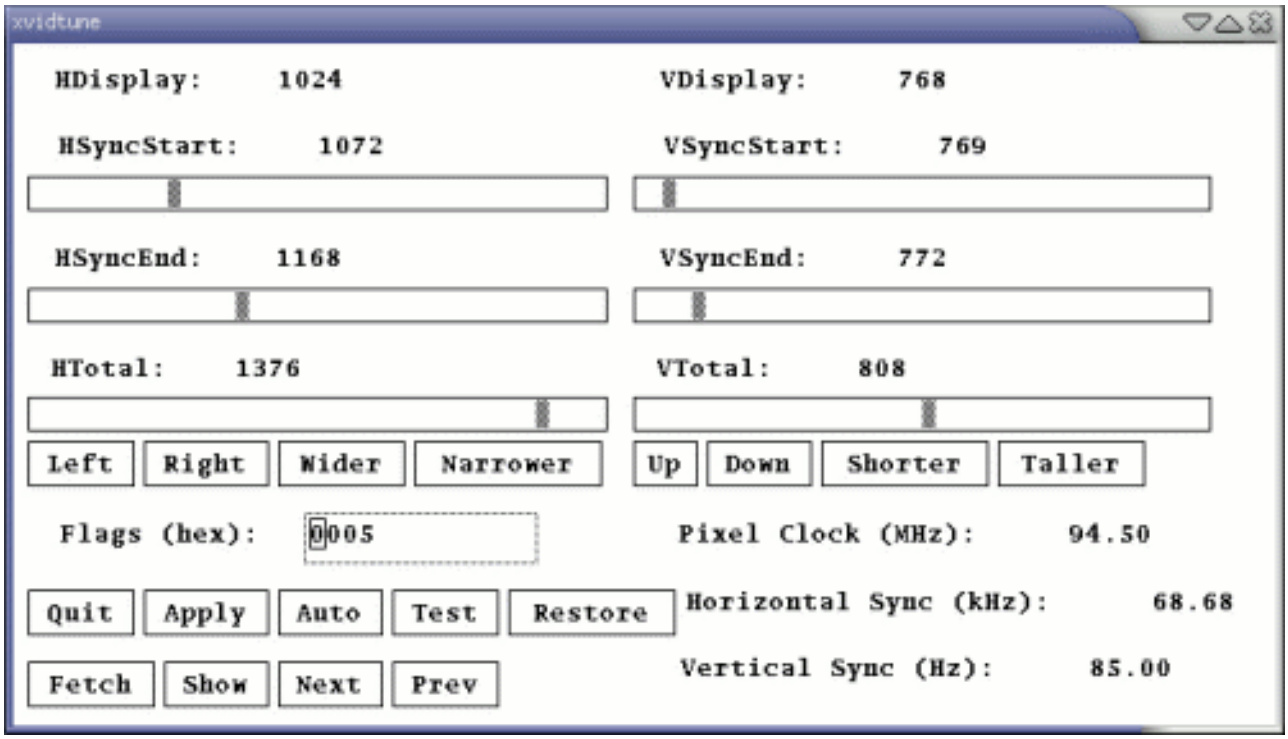
Dotaz. Prosím o radu, kterak vypočítat modeline pro rozlišení 1024x768 při 75 Hz. Při použití XFree 3.3.6 mi to jde bez problému samo, v XFree 4.3.0 mi však obraz zhasíná a problikává.

Odpověď. Nejlepší je použít některou z webových kalkulaček.

- http://xtiming.sourceforge.net/cgi-bin/xtiming.pl
- http://koala.ilog.fr/cgi-bin/nph-colas-modelines
- http://www.hut.fi/Misc/Electronics/faq/vga2rgb/calc.html
- http://www.dkfz-heidelberg.de/spec/linux/modeline/
- http://www.dynaweb.hu/opensource/videogen/

Dalším řešením je prozkoumání souboru `/var/log/XFree.0.log`. Jsou v něm informace z běhu serveru. Mezi nimi je také možné nalézt modelines, které nebyly z nějakého důvodu použity. Občas lze najít i ten řádek, který můžeme využít.

Třetí příklad, který mě napadá, je utilita xvidtune, s jejíž pomocí je možné modelines nastavit. Je to činnost na starším hardwaru potenciálně nebezpečná a celkově nepřehledná. Spusťte program z emulátoru terminálu (xterm); do něj se budou požadované informace vypisovat.



Chybové zprávy

Dotaz. Po spuštění XFree mi systém vypíše nějaké chybové hlášky, ale nestačím je při startu přečíst. Zapisují se někam do logu?

Odpověď. Ano, najdete je v již zmiňovaném souboru `/var/log/Xfree.0.log`. Při spuštění X serveru je stávající soubor přejmenován, obvykle získá příponu `.old`.

V některých systémech existuje ještě soubor `.xsession-errors`, který se nachází v domovském adresáři uživatele a obsahuje chybová hlášení procesů, které spouští *uživatel* při startu X serveru.

Automatické spouštění XFree

Dotaz. Chtěl bych, aby se mi XFree s KDE spustilo hned při startu. Abych nemusel pokaždé psát `startx`. Nikde jsem nenašel tuto volbu. Asi jsem blbě hledal, můžete mi helpnout?

Odpověď. Existují v podstatě dva způsoby jak spustit grafický server.

Příkazem `startx` z terminálu. Musí se to udělat ručně a spustíme tak pouze vybrané prostředí (KDE, GNOME, fluxbox, xfce nebo libovolné jiné).

Pomocí *runlevelu* neboli úrovně běhu systému. Systém se vždy nachází v některém runlevelu. Jeden je při načítání, další je pro přechod do víceuživatelského, nebo jednouživatelského režimu, další je pro podporu síťového prostředí, jiný je pro ukončení běhu systému. Existuje také runlevel pro grafické prostředí. Pro každou úroveň jsou v konfiguračních souborech stanoveny programy a služby, které se mají spustit, případně ukončit.

```
# Takto jsou runlevely definovány ve Slackware
# 0 = zastavení systému
# 1 = jednouživatelský (správcovský) režim
# 2 = nepoužito, stejné jako 3
# 3 = víceuživatelský režim, standard
# 4 = spuštění XFree se správcem přihlášení XDM, KDM, nebo GDM
# 5 = nepoužito, stejné jako 3
# 6 = reboot
#
#######################################################
#
# Takto jsou runlevely definovány v SUSE 9.1
#
# runlevel 0 je zastavení systému - nepoužívat jako standard!
# runlevel 1 je jednouživatelský režim
# runlevel 2 je víceuživatelský režim bez podpory sítě
# runlevel 3 je víceuživatelský režim s podporou sítě
# runlevel 4 není použit
# runlevel 5 je víceuživatelský režim s podporou sítě a XFree
# runlevel 6 je restart systému - nepoužívat jako standard!
```

Pro XFree je to runlevel číslo 4, v některých distribucích 5. Příkaz pro manuální změnu runlevelu je `telinit cislo-runlevelu`. Takto může správce systému (root) manuálně měnit stav, ve kterém se systém nachází.

Pro nastavení průběhu startu systému existuje soubor `/etc/inittab`. V něm najdeme řádek

```
# Default runlevel. (Do not set to 0 or 6)
id:4:initdefault:
```

Číslo stanoví runlevel, do kterého se systém přepne po dokončení startovacích rutin. Uvedený příklad je pro Slackware. Dvojtečky jsou důležité!

Jak shodit XFree?

Dotaz. Ahoj, jak mám shodit XFree server? Potřebuju nainstalovat nvidia ovladače, a nevím jak shodit XFree.

Odpověď. Pokud máte X server spuštěný výše uvedeným způsobem, jediná možnost obvykle bývá přepnout se na některou konzoli (na třetí např. `Ctrl-Alt-F3`), přihlásit jako root a zadat příkaz pro přepnutí do víceuživatelského režimu bez grafického prostředí. Tedy ve Slackwaru např. příkazem `telinit 3`. Po provedení změn v konfiguraci XFree se vrátíte do čtvrtého runlevelu příkazem `telinit 4`.

Pokud nemáte v konfiguračním souboru zakázanou klávesovou zkratku (parametr `DontZap`), lze X server shodit kombinací `Ctrl-Alt-Backspace`. Více informací najdete v popisu sekce `ServerFlags` v prvním díle tohoto seriálu.

Tento postup je ale účinný pouze v případě, že jste X server spustili příkazem `startx`; pouze v tomto případě se okamžitě dostanete do terminálu. Pokud používáte správce přihlášení XDM, KDM nebo GDM, celý grafický systém se *pouze restartuje* a znovu se zobrazí nabídka k přihlášení. Z toho vyplývá, že můžete (za běhu X) provést změny, přepnout se do X a klávesovou kombinací `Ctrl-Alt-Backspace` je restartovat. **Jde opravdu o tvrdý restart, žádná data se neukládají, tak si je nezapomeňte uložit!**

Server se znovu spustí a načte již novou konfiguraci. (Neplatí pro font server, ten je nutné restarovat ručně. To je ale nutné pouze v případě, že provádíte změny v jeho nastavení.)

Instalace karty

Dotaz. Potřebuji nainstalovat ovladače grafické karty, ale nevím kde začít. Instalace přiřadila kartu do generické skupiny "VESA". Přesné označení typu to asi nebude, asi spíš výrobce. Bohužel ani nevím, jak ty informace získat (tištěné materiály nemám k dispozici). Lze tu kartu detekovat automaticky? Půjde vůbec rozchodit?

Program lspci

Odpověď. Začít bychom měli utilitou `lspci`, která zobrazí zařízení sběrnice.

```
# lspci
00:00.0 Host bridge: VIA Technologies, Inc. VT8366/A/7 [Apollo
KT266/A/333]
00:01.0 PCI bridge: VIA Technologies, Inc. VT8366/A/7 [Apollo
KT266/A/333 AGP]
00:0a.0 Multimedia video controller: Brooktree Corporation Bt878
Video Capture (rev 11)
00:0a.1 Multimedia controller: Brooktree Corporation Bt878 Audio
Capture (rev 11)
```

```
00:0b.0 Multimedia audio controller: Ensoniq ES1371 [AudioPCI-97]
(rev 06)
00:10.0 USB Controller: VIA Technologies, Inc. USB (rev 80)
00:10.1 USB Controller: VIA Technologies, Inc. USB (rev 80)
00:10.2 USB Controller: VIA Technologies, Inc. USB (rev 80)
00:10.3 USB Controller: VIA Technologies, Inc. USB 2.0 (rev 82)
00:11.0 ISA bridge: VIA Technologies, Inc. VT8235 ISA Bridge
00:11.1 IDE interface: VIA Technologies, Inc. VT82C586/B/686A/B
PIPC Bus Master IDE (rev 06)
00:12.0 Ethernet controller: VIA Technologies, Inc. VT6102 [Rhine-
II] (rev 74)
01:00.0 VGA compatible controller: ATI Technologies Inc Radeon
RV100 QY [Radeon 7000/VE]
```

Jak je vidět z výpisu, grafická karta má `BusID 1:00.0`. To je také první parametr, který zapíšeme ve formě

```
PCI:sběrnice:zařízení:funkce
```

do sekce `Device`. Hodnota `PCI:1:0:0` odpovídá v tomto případě AGP kartě.

```
Section "Device"
 Identifier "radeon0"
 Driver "radeon" # volitelne "ati"
 VendorName "ATI Radeon VE"
 Option "AGPMode" "4"
 Option "ForcePCIMode" "on"
 Option "AGPFastWrite" "on"
 BusID "PCI:1:0:0"
EndSection
```

Nálepka

Nepodceňujme také nálepky na samotném zařízení, mnohdy pro identifikaci postačují. Jen je nutné stroj rozdělat a trošku se potrápit při vytahování karty a dýchání x let staré prachové usazeniny...

Samotný X server

Samotný X server má také metody pro detekci hardwaru - nejsou ale tak spolehlivé jaké první řešení.

```
X -scanpci nebo X -probeonly
```

Poté již následuje pouze hledání na Internetu a zkoušení různých kombinací. Režim VESA běží, pokud je mi známo, pouze v pevně stanovených frekvencích; pokud je váš monitor nepodporuje, bývá právě tohle jádrem problému. Je třeba experimentovat. Tohle se ale týká pouze starých karet, nové takové problémy nemívají.

Změna rozlišení za běhu

Dotaz. Nevíte někdo jak změnit rozlišení v Xkách (ne jenom velikost zobrazované plochy) pomocí nějakého programu? Mám na mysli něco jako je ve Win98 ikonka vpravo dole, na kterou když poklepu, rozbalí se nabídka možných rozlišení a když na jedno z nich kliknu, rozlišení se změní, tedy bez toho, abych musel editovat konfigurak Xek.

Odpověď. Pokud máte v konfiguračním souboru definováno více režimů a povoleny klávesové zkratky, lze rozlišení přepínat pomocí `Ctrl-Alt-šedé plus` a `Ctrl-Alt-šedé mínus`. Více v prvním dílu.

Existuje ale také rozšíření X serveru (načítá se obvykle automaticky), které umožňuje přepínání podle možností karty (není třeba režimy explicitně definovat). Nazývá se `XRandr`, jedná se o relativní novniku, které ještě nemá žádoucí podporu a není stoprocentně funkční ve všech kombinacích. Nové KDE nebo GNOME jej však samozřejmě podporují.

Někdy musí nastat konec

Problematice bychom se mohli věnovat ještě dlouho, právě proto máme diskuze. Já jen doufám, že jsem tímto shrnutím pomohl někomu zkrátit čas při nastavování grafického systému Linuxu.

Souborové systémy - často kladené otázky

Kde najít soubory s logy od programů?

Většina programů zapisuje výstup chyb a podobně do souborů v /var/log/* a někdy i na konzoli, pokud je to program pro X ka a potřebujeme vědět co vypisuje zkuste ho spustit v xtermu nebo jiném terminálu.

Jak optimalizovat ext3?

Nejjednodušší je změnit parametry mountování v souboru `/etc/fstab`:

```
noatime
```

Nezapisuje čas čtení souborů.

```
commit=60
```

Změny provedené ve FS se budou zapisovat na disk nejdéle po uplynutí jedné minuty (defaultně 5s) - do commitu zůstávají v diskové cache v RAM.

Řádek v `/etc/fstab`, pak bude vypadat např:

```
/dev/hdXX /mnt/sklad ext3 defaults,noatime,commit=60
```

Poněkud zásadnější změnou je nastavení žurnálování na `data_writeback`. I to lze provést v `/etc/fstab`, ale doporučuji použít příkaz:

```
tune2fs -o journal_data_writeback /dev/hdXX
```

K uplatnění těchto změn stačí remount.

Nakonec nastavíme vlastnost ext3 `dir_index`. Stručně: ke každému adresáři vytvoří jakýsi index (hashed b-tree), díky kterému je procházení velkých adresářů rychlejší.

```
tune2fs -O dir_index /dev/hdXX
```

Nutné je disk unmountnout, provést příkaz `tune2fs` a pak nechat vytvořit strom příkazem:

```
fsck.ext3 -Df /dev/hdXX
```

Pozn: je možné, že je tato vlastnost už aktivní. Zkontrolovat to můžete pomocí příkazu

```
tune2fs -l /dev/hdXX | grep features
```

```
Filesystem features: has_journal ext_attr resize_inode dir_index
filetype needs_recovery sparse_super large_file
```

Opakovaná aktivace nebo použití příkazu `fsck.ext3 -Df` není nebezpečné. Všechny tyto akce je nutné provádět jako root.

Jak namountovat .iso obraz jako souborový systém?

```
mount -t iso9660 -o loop jmenoobrazu.iso
/mount/adresakamhomountuji/
```

PS: Funguje to i na jiné (např. '.mdf ')

Jaký je rozdíl mezi ReiserFS a Ext3?

Reiser byl první žurnálovací FS pro linux.

Rozdíl je v implementaci souborového systému, rychlosti a taktéž ve stabilitě. Jinak na server bych Vám doporučil souborový sytém XFS... Jedná se o VYSOCE stabilní, žurnálovací a rychlý souborový systém.

Ext 2/3 jede ve všech verzích linuxového jádra za posledních (nejméně) šest let (ext3 ne), dokonce se k němu dá dostat i v jiných operačních systémech (nejen třeba FreeBSD, ale i Windows).

Tyto odpovědi jsou "výcucem" (copy/paste) z diskuze na ABCLinuxu.

Vidím špatně diakritiku ve jménech souborů

Tato otázka je zodpovězena ve článku Na co se často ptáme: /etc/fstab.

Jak mohu změnit velikost diskového oddílu bez formátování?

Tuto funkci nabízí například GNU Parted nebo některé komerční programy. Obecně se ovšem vždy vyplatí před takovým pokusem zálohovat data.

Rovněž může být výhodné využít LVM, které následně umožní diskové oddíly velmi snadno vytvářet, rušit a měnit jejich velikost. Umožňuje i mít oddíl na více discích. Nicméně oddíly musí být do LVM přiřazeny předem.

Je možné vrátit změny provedené na žurnálovacím souborovém sytému?

Nikoliv, žurnálování je metoda, která má za úkol zabezpečit konzistenci souborového systému pro případ náhlého výpadku. Data jsou v žurnálu uchováváno pouze po velmi krátkou dobu a nejsou nijak archivována. Nepostačují tedy ke vrácení změn.

Chci konvertovat souborový systém, aniž bych ho musel formátovat

V některých případech to možné je. Například souborový systém ext2 lze snadno povýšit na ext3:

```
# tune2fs -j /dev/hda1
```

Toto je dáno podobností těchto souborových systémů (jsou v zásadě stejné, ext3 navíc obsahuje podporu žurnálování).

Většina souborových systémů je ale natolik odlišná, že je snadno převést nelze. V takových případech musíme buď vytvořit zálohu, naformátovat disk a obnovit na něj zálohu nebo použít některý dostupný komerční produkt, který takové konverze umí, jako je např. Partition Magic.

Jak připojit FAT oddíl?

Souborový systém FAT je pod Linuxem podporován pro čtení i pro zápis. Existují různé verze FAT systému podle délky indexů (ovlivňují mimo jiné maximální velikost oddílu), označují se FAT12, FAT16 a FAT32. Tento souborový systém připojíte příkazem mount

```
# mount -t msdos /dev/fd0 /mnt/floppy
```

Tento příkaz, provedený jako superuživatel „root“ připojí obsah diskety do adresáře `/mnt/floppy`, který musí existovat už před provedením příkazu.

Typ souborového systému (`msdos`) lze mnohdy vynechat, nebo zadat `auto`.

Krom typu `msdos` existuje typ `vfat`, který se liší podporou dlouhých názvů souborů (používají Windows). `vfat` je zcela zpětně kompatibilní.

Smazal jsem důležitá data, jak je mohu obnovit?

Důležité je, na jakém souborovém systému se data nacházela. Obecně je nutné přestat na daný souborový systém dále cokoliv zapisovat (můžete ho nejlépe odpojit, nebo připojit pouze pro čtení).

Pro souborový systém `ext2` nabízí tuto funkci souborový manažer `mc` (viz jeho manuálové stránky), utilitu unrm. Vyzkoušet můžete nástroj testdisk. Před podobnými pokusy je rozumné provést zálohu oddílu.

Pro souborový systém reiserfs je obnova často nemožná. Můžete vyzkoušet tento postup (velmi důrazně doporučuji zálohu souborového systému!).

Pokud jsou data velmi cenná, můžete se obrátit na specializované společnosti, ovšem ceny za obnovu dat jsou velmi vysoké.

Jak připojovat souborové systémy jako běžný uživatel?

Aby bylo možné připojit souborový systém jinak než jako superuživatel, je nutné pro něj vytvořit záznam v konfiguračním souboru /etc/fstab. Jeho podoba je popsána v článku Na co se často ptáme: /etc/fstab.

Pro možnost připojení jakou běžný uživatel je důležitý parametr `user`. U zařízení jako CD mechanika nebo disketová mechanika budete ještě nejspíše chtít uvést parametr `noauto`, který určí, že se systém nebude pokoušet zařízení připojit při startu.

Zde je příklad pro připojování flash disku a CD/DVD mechaniky.

```
/dev/sda    /flash  auto     noauto,user     0 0
/dev/cdrom  /cdrom  iso9660  noauto,user,ro  0 0
```

U CD mechaniky parametr `ro` udává, že zařízení bude připojeno pouze pro čtení.

Následně je možné připojovat souborový systém příkazem `mount`.

```
# mount /mnt/cdrom
```

a odpojit

```
# umount /mnt/cdrom
```

K připojování a odpojování je možné využít i grafické nástroje, které jsou často součástí Desktopového prostředí.

Tento postup je obecně platný pro všechny distribuce, mnohé novější distribuce přidávají záznamy do fstab automaticky nebo pomocí konfiguračních nástrojů.

Mohu použít přístupová práva pod FAT oddílem?

FAT oddíl neumožňuje uložit unixová přístupová práva (není pro to navržen). Veškeré změny v právech na tomto souborovém systému budou po jeho odpojení ztraceny, protože je není kam uložit.

Pro všechny soubory v FAT souborovém systému se použijí práva nadefinovaná v fstab nebo určená parametrem `mount`. Konkrétně jde o pravidlo `umask`, které nastavuje masku práv, ve tvaru `umask=022` (co konkrétní hodnoty znamenají najdete ve slovníku pod pojmem Práva).

Pokud přesto chcete využívat přístupová práva pod FAT souborovým systémem, můžete využít souborový systém `umsdos`, který práva zaznamenává do vyhrazených souborů.

Jak mohu připojit NTFS disk, jaká je podpora NTFS pod Linuxem?

Připojení disku s NTFS s využitím jaderného modulu `ntfs`

Modul pro práci s NTFS (ntfs.ko) který se vyskytuje v distribučních jádrech umožnuje pouze jeho čtení. Pokud je tento modul zkompilován s podporou pro zápis, tak pozor na jeho omezení - nelze měnit názvy souborů ani vytvářet soubory nové. Chcete-li tedy nějaký soubor přesunout na disk s NTFS, pak jej musíte nakopírovat do již existujícího souboru. Připojit diskový oddíl s NTFS můžete zcela běžným způsobem pomocí příkazu `mount`.

```
# mount -t ntfs /dev/hda2 /mnt/disk_c -o nls=utf8
```

Připojí druhý primární oddíl na prvním IDE disku do adresáře `/mnt/disk_c` (te před použitím příkazu `mount` musí existovat a měl by být prázdný - po připojení diskového oddílu se totiž k jeho obsahu nedostanete, dokud jej neodpojíte).

Pokud chcete k tomuto diskovému oddílu přistupovat i jako běžný uživatel, přidejte do parametru "-o" hodnotu *umask="vaše_uid"*. Všem uživatelům povolíte přístup hodnotou *umask=000*.

Aby nebylo nutné připojovat disk manuálně po každém restartu, přidejte řádek s příslušně upravenými hodnotami do souboru `/etc/fstab`

```
/dev/hda2 /mnt/disk_c ntfs auto,nls=utf8,umask=000   0   0
```

Více man fstab.

Kompletní podporu NTFS (čtení i zápis) poskytuje řešení přes ntfsprogs s využitím FUSE. Modul pro FUSE (fuse.ko) se vyskytuje v jádře od verze 2.6.14 (starší verze jádra je nutno patchovat), lze přes něj lokálně připojovat nejenom diskové oddíly s NTFS, ale také např. vzdálené adresáře přes SSH, FTP, WebDAV, atd..

Připojení disku s NTFS s využitím jaderného modulu `fuse`

Nejprve si ověřte, obsahuje-li vaše distribuce balíky pro práci s fuse (`fuse-utils` a `libfuse2`) a ntfs (`libntfs8` a `ntfsprogs`). Podotýkám, že názvy balíků těchto nástrojů se mohou v jednotlivých distribucích lišit. Pak si zkontrolujte zda-li je máte nainstalovány ve vašem systému.

```
## V balíčkovacím systému postaveném na DEB
dpkg -l | grep "\(ntfs\|fuse\)"

## V balíčkovacím systému postaveném na RPM
rpm -qa | grep "\(ntfs\|fuse\)"

## V balíčkovacím systému Portage
epm -qa | grep -e ntfsprogs -e fuse # pokud nemáte epm, tak emerge epm
```

1. Nejprve je nutno zajistit, aby se modul `fuse` natahoval ihned při startu, takže přidáme jeho název do souboru `/etc/modules` (Ubuntu)

   ```
   echo fuse | tee -a /etc/modules
   ```

2. Pak vytvoříme skupinu `ntfs`

   ```
   addgroup ntfs
   ```

   Při vytváření skupiny by mělo vyléz něco podobného:

   ```
   Adding group `ntfs' (1001)...
   Done.
   ```

   Tu nastavíme všem uživatelům, co by měli mít zápis do diskového oddílu s NTFS povolen

   ```
   adduser uzivatel ntfs
   ```

3. Do souboru `/etc/fstab` přidáme patřičně upravený následující řádek

   ```
   /dev/hda1    /mnt/disk_c    ntfs-fuse    auto,gid=1001,umask=0002    0
   0
   ```

   Pokud chcete zablokovat přístup uživatelům co nejsou ve skupině `ntfs` tak nastavte `umask=0007`

Pozn.: V Ubuntu Dapper se vyskytuje bug. Je třeba opravit symlink `/sbin/mount.ntfs-fuse`

```
bash:~$ sudo rm /sbin/mount.ntfs-fuse && sudo ln /usr/bin/ntfsmount
/sbin/mount.ntfs-fuse
```

Po restartu by mělo být možné disk s NTFS nejen číst, ale také na něj zapisovat.

Zde odkaz na stránku s postupem nastavení ntfs-fuse ze které jsem čerpal.

Zápis na disk s NTFS přes `captive`

Podporu pro zápis nabízí také třeba projekt captive.

Dodatky k učebnici

Kernel

Kernel, nebo-li také jádro, je základním programem, který se zavádí po spuštění Linuxu. Mezi jeho hlavní úkoly patří ovládat veškerý hardware počítače, spravovat procesy a virtuální paměť a řídit přerušení.

Původně platila rovnice kernel = Linux. Tedy Linus Torvalds napsal jádro operačního systému, které konzistentně **obaluje hardware** a poskytuje základní stavební kameny pro aplikace (**userland**), ale samo o sobe nepřináší uživateli zadny primy užitek.
Po spojeni s projektem **GNU** (GNU's not UNIX) Richarda Stallmana se začalo pod pojmem Linux označovat **cele prostredi** včetně aplikaci (tedy Linux = kernel + GNU). Vydávání Linuxu se pote chopili různí **distributoři**, kteří do kernelu i programu z projektu GNU začali přidávat svá (místy komerční /uzavřená) **rozšíření**. Situaci vystihuje označováni jména distribuce Debian GNU/Linux (tedy Debian = kernel + GNU + případně vlastni úpravy distributora).

Existuje tzv. **vanilla kernel** ("od Linuse"), který mel až do verze 2.6.8 číslováni ve tvaru MAJOR.MINOR.SUB (lichá MINOR jsou vývojové verze), později ve tvaru MAJOR.MINOR.SUB.FIX (vsechna MINOR jsou stable, FIX jsou bugfixy posledního SUB) a **odvozene verze** (defacto patchsety) významných vývojářů, které jsou od vanilly odvozené. Poznají se podle toho, ze konci pomlčkou a dvěma písmeny: -ac (Alan Cox), -mm (Andrew Morton) apod. Tyto upravené verze pak obsahuji změny, které se ještě nedostaly do **hlavního stromu jádra**, ale mohou byt užitečné pro spoustu uživatelů. Vice viz. http://kernel.org/pub/linux/docs/lkml/ a http://en.wikipedia.org/wiki/Linux_kernel

GNU

Rekurzivní akronym pro GNU is Not Unix, kde GNU znamená GNU is Not Unix, kde ...

Projekt založený v roce 1984 Richardem Stallmanem a společností Free Software Foundation, na vytvoření svobodného a otevřeného operačního systému (dále jen OS) na základě OS UNIX. Zpočátku soubor systémových programů, kterým chyběla hlavní součást - jádro (kernel). Později se jako jádro použil projekt Linuse Torvaldse Linux. V mínění veřejnosti se společný OS GNU/Linux přejmenoval pouze na Linux podle jádra. Jediná distibuce , která dodržuje správné pojmenování tohoto OS je Debian GNU/Linux.

RMS

Richard **M**atthew **S**tallman je zakladatelem hnutí Svobodného softwaru, projektu GNU (v jehož rámci napsal množství významných softwarových balíků, např. překladač gcc, editor Emacs a další) a zastřešující nadace FSF. Je rovněž autorem licence GNU GPL.

V současné době se RMS věnuje celosvětové propagaci myšlenek svobody jednotlivce, především pak na poli počítačových programů.

Články na abclinuxu.cz:
- Rozhovor: Richard Stallman
- Křížová výprava Richarda Stallmana za svobodou
- Richard Stallman v Praze!
- Komix: RMS v Praze
- Co je to Linux

Copyleft

Copyleft je metodou (použitou v GNU GPL), která znemožňuje přeměnu svobodného softwaru na software proprietární (nesvobodný).

Hlavní myšlenkou copyleftu je dát každému povolení ke spouštění, kopírování, modifikaci programu a šíření modifikovaných verzí, ne však povolení přidávat k nim vlastní omezení. Takto jsou rozhodující svobody, které definují svobodný software zaručeny pro každého, kdo má kopii; stávají se nezcizitelnými právy.

Aby byl copyleft efektivní, musí být modifikované verze rovněž svobodné. To zajišťuje, že naše práce je, pokud je zveřejněna, dostupná naší komunitě.

Zdroj, URL: http://www.gnu.org/gnu/thegnuproject.cs.html

GNU GPL

Pod GNU's Not Unix General Public License je vydávána většina Linuxových programů a distribucí. Hlavní myšlenkou tohoto projektu je vývoj tzv. otevřeného(svobodného) software, tj. takového, ke kterému jsou dostupné zdrojové kódy a je zdarma. (http://www.gnu.org)

Český překlad GNU GPL je možné nalézt na www.gnu.cz (spolu s překlady dalších GNU licencí), anglický originál tamtéž nebo na www.gnu.org, navíc je přiložen ke každém GPL programu jako `COPYING`.

Zdrojový kód ke GPL programům je nejen dostupný, ale mohu jej libovolně modifikovat a šířit dále nebo použít v jiných programech -- za předpokladu, že zachovám původní copyright a že výsledek je opět licencovaný pod GNU GPL.

GNU GPL (GNU GENERAL PUBLIC LICENSE) je softwarovou licencí, která se hlavně zabývá otázkou přístupu ke zdrojovému kódu softwaru a otázkou redistribuce tohoto zdrojového kódu. Je navržena tak, aby autor měl povinnost zpřístupnit zdrojový kód programu. Uživatel zase může daný zdrojový kód s modifikacemi nebo bez nich dále šířit zase jenom pod touto licencí.

Každý program pokrytý GNU GPL je svobodný. Věta obrácená není pravdivá.

HURD

Jadro operačného systému GNU. HURD je mikro-kernel, čo znamená veľkú flexibilitu, ale obtiažny vývoj. Viac o HURDe najdete na http://www.gnu.org/software/hurd/

X Window System

XFree neboli X Window System je grafické prostředí, které **může** být spuštěno po startu systému. Není však vůbec nezbytné a některé distribuce, zejména serverové nebo speciální, jej zcela postrádají.

Hlavním řídicím konfiguračním souborem je `/etc/X11/XF86Config`.

Články na abclinuxu.cz:
- X Window System - I
- X Window System - I (Monitory, grafické režimy a modelines)
- X Window System - III (Praktická řešení)

Jinými slovy, **XFree** (neboli **X server**) obstarává v Linuxu grafické prostředí (tedy to, bez čeho si většina lidí neumí počítač představit ;o)

Pokud **X server** není spuštěn (případně vůbec není nainstalován), nabízí Linux práci pouze ve znakovém režimu v tzv. "terminálu". V běžícím grafickém prostředí jej emuluje tzv. "konsole". Konsole vypadá a funguje jako "příkazový řádek" v M$ Windows, ale nabízí neskonale větší množství možností.

(Někteří ortodoxní sysadmini odmítají pracovat v čemkoli jiném a X server považují za tragický omyl a chybný krok ve vývoji Linuxu. ;o)

KDE

K Desktop Enviroment. Správce oken a balík základních aplikací (browser Konqueror, e-mail client KMail, kancelářský balík KOffice, emulátor terminálu Konsole...) postavený na knihovně QT firmy Trolltech.

KDE, podobně jako Gnome nebo XFce, není pouze správce oken (WM - window manager), ale kompletní desktopové prostředí. Správce oken je jednou z komponent KDE.

Gimp

GNU **I**mage **M**anipulation **P**rogram je svobodný grafický bitmapový editor. Jeho prvotní verze používaly toolkit Motif, ale kvůli licenčním problémům došlo k napsání nového toolkitu gtk (**G**imp **T**ool**k**it), který dnes používá spousta opensource softwaru.

Byl původně zamýšlen jako náhrada Adobe Photoshopu. Tohoto cíle se nepodařilo úplně dosáhnout, nicméně pro většinu lidí představuje plnohodnotnou náhradu. Týká se to především technologií, které mají nesvobodnou licenci, která neumožňuje jejich implementaci pod GPL (Panetone, ...).

Maskot Gimpu se jmenuje Wilber.

Jak řešit problémy

Když procházím občas konference a hledám řešení nějakého problému, objevím tam dotazy linuxových nováčků. Některé dotazy jsou typu "mě to nefunguje a nevím co s tím", "proč nejde zařízení či soft pod linuxem", a podobně. U některých dotazu je znát, že autor nepřečetl ani kousek dokumentace a ani nic neudělal pro to, aby se pokusil dotazovaný problém alespoň trochu sám řešit. Linuxoví guru ani na tyto typy dotazů neodpovídají či odporučí přečtení README souboru či manuálové stranky.
Na jednu stranu se jim ani nedivím, když autor dotazu pro řešení problému ani nic neudělal. Na druhou stranu si říkám, že nováčkové ani občas nevědí, jak řešit dané problémy či stráví neefektivní prací dlouhé hodiny. Článek by měl pomoci alespoň trochu začátečníkům a pro ty, co už něco znají, by mohl ušetřit čas na odepisování do konference či sledování dotazů, na kterých nebylo znát ani trochu snahy jejich autora. Nováčkové by si měli uvědomit, že čím více času věnují oni danému problému, tím více se mohou věnovat guru vývoji software, který může linuxové komunitě a i jim pomoci. Na druhou stranu chápu a ctím to, že každý problém má svůj čas na řešení a strávit nad něčím hodiny bez výsledků nemá smysl.

A tak jsem se rozhodl napsat jakýsi manuál, jak postupovat při řešení problémů co nejefektivněji. Nepočitejte s návodem jako z knížky, je to pouze popis pár mých praktických zkušeností. Je spíše určen pro ty, co s linuxem začínají. Od guru ocením, když napíší nějaká podobná řešení na toto téma či mne zkritizují. Sám se necítím jako guru ani jako znalec. Používám linux asi 2 roky občasně a poslední půlrok denně 6-14 hodin. Tomu, co umím, vděčím právě čtení dokumentace, rad od kolegů, čtení konference. Žádná věda. A je za tím mnoho práce a času.

Toto téma je bráno jako volné, nechci zde řešit příliš nějaké konkretní problémy, spíše jenom občas příklady. A ani v diskusi pod článkem. Od toho je tu nové diskusní fórum. V diskusi by neškodilo, kdyby guru přidali něco ze svých zkušeností. Předem díky. Problémy jsou řešeny částečně dle obtížnosti do bodů, ale příliš na jejich pořadí nehledím. Vemte si z článku každý, co potřebujete.

Nastal problém. To či ono nejde. Co s tím?

0) Literatura
Pokud do linuxu vůbec nevidíte, doporučuju přečíst Linux-dokumentačni projekt. Ne celý. To co potřebujete vědet a myslíte si, že to užijete v praxi. Ať už knížku či lépe v elektronické podobě. Existují i školičky linuxu. Dobré linky jsou www.ll.cz , www.manualy.sk a sekce UNIX, www.penguin.cz. Kvalitní články jsou také na www.root.cz. Mnoho dokumentace v angličtině je na serveru linuxdoc.org. Samozřejmě existuje i více serverů, kde lze najít linuxové informace.
1) Samostatnost
Nepište hned do diskusního fóra či konference !!! Porvěte se s tím chvíli sami.
2) Man, Info
V témeř každém balíčku programu či distribuci existuje manuál. A určitě by jste se měli s ním dostat dále než na původní pozici a kontaktujte support@<vložte jméno jedné nejmenované firmy>.com .
Zapamatujte si příkazy `man` a `info`. U daného příkazu stačí většinou napsat `man <něco> nebo info <něco>`.

Hlavní pohyb po manualové stránce man je pomocí šipek, page_up, page_down, home, end, vyhledavání slov je pomocí znaku /. Hlavní pohyb po manualové stránce pomocí info je obstaravají `šipky, page_up, page_down, home, end, u -nahoru, n -další, p -předchozí, vyhledavaní slov znak /` . Vyhledavání pomocí manuálových stránek je možno pomocí man (-k, -K, případně jiné volby z man man).

3) Instalace programů
Instalovat programy doporučuji z rpm či deb balíčků či dle balíčků dané disribuce. Máte alespoň pořádek na disku a vyznáte se v tom. Je také lepší možnost odinstalace. Někdy není na zbytí a daný

balíček je ve formátu zdrojových kódu anebo rpm či deb balíček nechodí či nejde nainstalovat. Ve většině balíčku (`tar.gz, tar.bz2`) jsou soubory `INSTALL, READ, INSTALL_INSTRUCTION, README_FIRST`, či adresáře `/doc, /INSTALL, /documentation`, neškodí si je pročíst. A nerad bych zapomněl na dokumentaci `HOWTO`. Alespoň lehce přečíst po klíčových slovech, které jsou samostatně na řádku. Hlavní slova jsou `./configure, make, make install`.

Ne vždy jde instalace pouze pomocí `./configure, make, make install` a je hotovo. Zvažte také, zda se vyplatí upgradovat. Tyto informace najdete opvykle v souborech Changelog či Changes. Může také pomoci `./configure --help` pro nastaveni instalačních a kompilačních voleb programu.

Pokud spouštíte programy, tak nápovědu či volby programu lze získat pomocí příkazu `<jméno_programu>` (a zkuste připojit jednu z voleb) `-h --h -help --help` (někdy stačí napsat samotné jméno programu.

4) Jazyky
Neškodí znát trochu angličtinu. Nemusíte proto navštěvovat nějaký kurs či nosit s sebou slovník. Stačí si občas zapamatovat nějaký termín. Dobrou pomůckou může být na vedlejší konsoli otevřený v linksu či jiném prohlížeči online slovnik, osobně používám slovnik.seznam.cz.
5) Textový režim
Nebojte se používat textový režim, ať už konsoli či v X. Používám jej především pro to, že je rychlejší než grafický režim a především, že hledání na internetu v linksu je nesrovnatelně rychlejší než stahování reklam, bannerů a všelijakých grafických hračiček pomocí Mozilly či jiných prohlížečů v X-kách.

Grafické hračky a myš hrozně zdržuji. Hlavně myš. Pokud se naučíte používat klávesnici a klávesové zkratky, zjistíte, že myš je například dobré praktické těžítko, aby se samy nezavíraly stránky v knížce, z niž něco studujete a že občas se s nic dá i pracovat. Někde je myš ale nutnost, to nepopírám.

6) Rozšiřte si pracovní plochu
Používejte více konsolí či obrazovek. Od toho tam jsou. Šetřit papírem je logické, ale obrazovkou ne. Ale zase ne na úkor přehlednosti. Přepínání konsolí: `Alt+F1 až Alt+F6`. `Alt+F1 až Alt+F4` přepínání pracovních ploch X. Konzolí a pracovních ploch si navolte, kolik chcete a kolik vám linux dovolí.
7) Konference
Používejte archívy konferencí. Osobně doporučuji linux@linux.cz. Někdy stačí projít mailing-listy daného sw či hw a dotaz z problémem tam byl obvykle již položen.
8) Vyhledávače
Kámoš google. Existuji i jiné vyhledávače(webfast, yahoo, seznam). Google je moc chytrej vyhledávač. Stačí vložit chybu či chybovou hlášku a on vám ukáže stránku s ní a možna je to další cesta k řešení, obvykle se dostanete na nějakou stránku projektu anebo do nějakého archívu konference. A problém, který řešíte, už někdo většinou vyřešil před vámi.
9) AbcLinuxu
Nemusíte chodit moc daleko. Někdy stačí AbcLinuxu anebo dát si vyhledat dané slovo na root.cz a v článcích (ne archív krátkých zpráv) něco najdete.
10) Freshmeat
Hledáte software? Mrkněte se na freshmeat. Super rozcestník, vyhledávač a katalog software, je ale anglicky.

11) Textové editory
Používejte rozumný editor textu. Já osobně jsem už pár měsíců stále fascinován editorem Vim a mám problémy psat v nějakém jiném editoru, protože práce mi potom příliš dlouho trvá. Ale existují i jiné editory. Každý by si měl z široké nabídky urcite vybrat. Emacs, joe, gedit, nedit, Koffice, editor v mc.
`echo slovo > soubor` - *pro ty opravdu tvrde linuxáře*

Osobně říkám, že správní muži píší ve Vim, dělají na konsoli a píší v noci. Sám bych ale k tomuto přívlastku potřeboval poněkud více znalostí ....

12) Grep
Mocný příkaz `grep`. Občas potřebujete najít nějaký termín či pojem. Máte před sebou haldu dokumentace a zdrojáků. Asi je nebudete číst všechny. Projeďte je grepem.
```
grep -air 'hledane_slovo_ci_vyraz'
soubory_mozno_s_hvezdickovou_syntaxi
```
`-a` jako text, `-w` slova, `-i` nerozlišovat malá a velká písmena `-r` rekursivně (třeba celý adresářový strom)

Dále neškodí použít příkazy `cut` a `sort`. Přečtěte si jejich manualové stránky. Fakt moc šikovné příkazy.

13) Roury
Propasírujte příkaz či výpis programu přes rouru. Je to dost schopný způsob filtrovaní informací.

Příklad: `příkaz | grep -air 'slovo' | sort`

14) Přesměrování
Přesměrujte si výstupy z programu. Získáte tím výpis chyb z obrazovky. Na obrazovce se lze vracet asi o 5 obrazovek zpět pomocí `Shift+Page_Up/Page_Down`, ale co když je toho více a chcete s tímto textem pracovat.

Příklady:

výpis souborů z adresáře do souboru
`ls -l > adresar_list_soubor` a můžete s tím hned pracovat
výstupy z kompilace
`make > message_file` a hned se ty chyby hledají lépe.

15) Logy
Většina větších a inteligentních programů zapisuje hlášky o své činnosti do určitých souborů = logů. Tímto stylem se dají nalezt chyby. Většina logů je v adresáři `/var/log/`. Případně lze zapnout tuto volbu u některých programů.
16) Ukecanost -- verbose
Mnoho programů má volbu `-v`. To je výpis o činnosti programu. Dá se zapnout i jeho úroveňn a množství. Z toho se dá potom i něco najít a případnou chybu propasírovat přes grep či googla. Když chybu nelze najít, tak jejímu objevení pomožte. Třeba i způsobit jinou chybu danou chybu doprovazející.
17) Zálohy a zápisy činnosti.
Když provádíte nějaké větší úpravy v systémových souborech a hrozí, že by se nerozběhl systém a že se budete muset vracet zpět, zálohujte si tyto soubory či používejte linky (příkaz ln). Neškodí si psát, co děláte. Ale na papír. Z neběžícího systému informace nedostanete. Vhodné to je také pro více správců serveru, aby se potom na serveru mohli lépe orientovat. Pokud se vám systém po vaší úpravě nerozběhne, tak není nic jednoduššího, než nabootovat z CDéčka či přenést disk a soubory ze zálohy překopírovat. Lepší než nová instalace systému a moře nervů a času pryč.

Doporučuji i zálohy konfiguračních souboru po delším čase na již vyladěném systému. Pak stačí pouze instalace systému a překopírování těchto souborů a ne pracné naklikávání a dopisování znovu. Není problém již v 35 minutě po instalaci na silnějším stroji pracovat na systému s většinou věci již nakonfigurovanými z přechozího vyladěného systému. Linux je sice dost stabilní systém, ale někdy stačí, když vám odejde harddisk a je o "radost" postaráno.

18) Neřešitelné problémy.
Někdy nemůzete něco vyřešit a trápíte se s tím dlouhou dobu. Linux není určen k tomu, aby se daný uživatel na něm dřel či z daného problému duševně zkolaboval. Je na vás, jak dlouho na daném problému pracujete, kolik máte času a zda jsou alespoň nějaké kroky a výsledky kupředu.

Diskusní fórum či konference. Od toho tu je. Pište stručně, ale výstižně. Snažte se maximálně pomoci co nejvíce těm, co vám chtějí pomoci. Nesdělujte učastníkům konference, ze &quit;daný OS je na <vložte nějaké neslušné slovo>", že "pod jiným OS tošlo bez problémů", šetřete jejich čas, poštu a celkově již ucpaný internet. Čekejte však, že dostanete návod, jak to či ono najít či řešit. Ne jak to přesně krok po kroku nainstalovat a zprovoznit. To by se rovnou mohl zadávat přístup na ssh a ten, kdo by zadával problém k řešeni by nemusel ani sáhnout na klávesnici a zároveň by se také nic nenaučil.

Pro mě je třeba neřešitelným problémem a noční můrou tiskárna Kyocera F1000A a appsfiltry. A po pár nocech s touto tiskárnou jsem se uz ptal v konferenci. Neříkám, že musíte něco zadat do konference až po několika dnech a probdělých nocech. Hlavní měřítko asi je, jak postupujete kupředu a zda jste dosáhli alespoň nejakých výsledku či ne.

19) Nákup HW
Ne vždy je dobré koupit HW za super levnou cenu či absolutní novinku. Hlavní měřítko je, zda to linux podporuje. Doporučuju stránky AbcLinuxu a Linux Hardware Database (je to database linuxem podporovaneho hardware). Či se podívat na stránky výrobce nebo na googla.

Ani HW zadarmo či jiné dárečky nemusí občas chodit. Nic proti prehistorickym kouskům, ale občas by to chtělo mít alespoň jistotu, jestli daný HW ještě "žije" a funguje, jak má. Ne-li zhodnotit, zda nejít místo zdlouhavé instalace pracovat a vydělat si na HW novější a o něco více funkční. Nejnovější a nejrychlejší "supervěc" zase nemusí být ještě podporována nebo pod Linuxem chodí na 50% výkonu.

20) Kompilace jádra a jeho instalace
Čtete, co dáváte do voleb. Je dobré si nechat minimálně jedno funkční jádro záložní. či mít bootovací disketu či CD s instalačkou. Potom můžete alespoň nabootovat. A ukladání předchozích konfiguračních souborů jádra `config` není od věci. Obzvláště pokud patříte mezi ty, co pravidelně updatují jadro a co chtějí podporu nových věci.

K update jádra. Přečtete si changelogy. Případně je spojte z více verzí do jednoho souboru a pak grepem zjistěte, zda update má smysl.

```
cat changelog* > all_changelog
grep -ai 'hledany_hw_k_update_jadra' all_changelogy
```

21)Instalace modulu
Jedno z řešení. Pokud Vám nejdou nahrát moduly k danému zařízení či HW zkuste příkaz `modinfo <nazev_modulu>`. Dostanete volby pro daný modul a potom můžete doplnit za `insmod options <dane_volby>`.
22) Pište scripty
Není potřeba vše pracně opisovat. Uložte si dané příkazy z příkazové řádky do souboru a ten spouštějte pomocí příkazu `sh`. Případně soubor s příkazy můžete na vedlejší konsoli editovat.

23) Zdrojáky
Trošku obtižnější. Někdy nejde něco přeložit. Či spustit. Nahlédněte do spouštěcích scriptů či do souborů `Makefile`. Občas stačí zeditovat cesty. Neškodí nějaká znalost C či jiného programovacího jazyku. Pomocí `locate` či `find` si najdete, zda daný soubor vůubec máte a kde ho máte.
24) Find
Pokud nenajdete soubory pomocí `locate` (z database) a nebo jste přidávali do systému nové soubory po update databáse nebo potřebujete vyhledávat podle určitých parametrů soubory, použijte `find`. Má neuvěřitelně mnoho voleb. Také mocný nástroj.
25) Regulární výrazy
Báječna věc. Doporučuju se je naučit, pokud děláte v linuxu více a myslíte to s ním opravdu vážne. Suprově slouží k vyhledávání daných slov, ať už v nějakém rozumnějším editoru či pomocí grepu. Ale neni to nic jednoduchého se je naučit. Pokud do nich proniknete, oceníte je jako nepostradatelné pomocníky a nahrazování a hledání bude o dost efektivnější. Věci typu najdi slovo s 5 znaky od konce řádku či nahraď každé čislo větší než 100 pěti hvězdičkami, budou potom běžností.

Více se ale podívejte na články pana `Satrapy` na www.root.cz (hledejte treba slovo "regular".

26) Root
Když nemusíte, nepracujte jako superuživatel `root`. Kdysi jsem si omylem potvrdil kompletní smazání adresáře `/usr/`. K tomu raději žádný komentář.
27) Zase konference
Doporučuju pravidelně pročítat. Myslím tím věci, které jsou použitelné a využitelné pro vás, ne do písmenka. Je to dobry vzdělávací prostředek.
28) Zdroje informací.
Vytvořte si vlastní síť informací na internetu. A podle toho, jak často na daných stránkách přibývají informace, tak tyto stránky navštěvujte. Někam stačí jít jednou za týden, někam každý den. To samé platí pro update software.

Přeju minimum problémů s linuxem a maximum nabytých znalostí.

BSD

Berkeley Software Distribution je verze Unixu vyvinutá na univerzitě Berkeley. Za tímto projektem stál Bill Joy, autor mnoha legendárních programů - editor ex, editor vi, C shell.

Hlavním konkurentem pro BSD bylo SystemV od AT&T. Na počátku 90. let byl uvolněn zdrojový kód a dnešní následovníci jsou FreeBSD, OpenBSD, NetBSD, NextStep a Darwin.

GNU LGPL

GNU Lesser General Public License, nebo také (starší název) GNU Library General Public License. Svobodná licence pro šíření softwaru, podobná licenci GNU GPL. Je určena k šíření dynamických sdílených knihoven.

Smyslem licence je poskytnout svobody zaručené licencí GPL (tedy neomezené užívání, přístup ke zdrojovému kódu, možnost modifikace, šíření původních i modifikovaných verzí vždy pod stejnou licencí), a současně umožnit použití v odlišně licencovaných programech.

Pro sdílené knihovny to má klíčový význam, protože existuje mnoho programů, které mají různé licence nekompatibilní s GPL - a tyto programy tím získají možnost sdílené knihovny zcela legálně používat. Současně je vlastní kód knihovny chráněn úplně stejně, jako kdyby byl šířen pod GPL.

Důležité je, že možnost použití v ne-GPL programech je omezena na dynamické přilinkování knihovny. Pokud by se kód knihovny do programu přilinkoval staticky, musely by se tento program vztahovat podmínky uložené licencí GPL.

Mach

Mach je mikrojádro na kterém je postaveno jádro HURD.

Linus Torvalds

Linus Benedict Torvalds zahájil vývoj jádra Linux a jeho hlavním maintainerem. Narodil se 28. prosince 1969. Linus se narodil v Helsinkách, Anně a Nilsovi Torvaldsovým.

Linus navštěvoval Helsinskou univerzitu v letech 1988 až 1996, graduoval s magisterským titulem z Informatiky. Svou diplomovou práci, nazvanou `Linux: A Portable Operating System` (Linux: přenositelný operační systém), napsal v Linuxu.

Zdroj: cs.wikipedia.org

Články na abclinuxu.cz:
- Co je to Linux

x86

x86 (zkrácenina od 80x86) je označení architektury procesorů, které vytvořila společnost Intel. Její název vychází z označení procesorů 8086, 80186, 80286, 386 (i386) a 486 (i486). Z obchodních důvodů přestal Intel označovat další řady číslicí, ale nové řady pojmenoval Pentium. Ty jsou označovány jako i586. Procesory Pentium Pro a II až IV jako i686.

Důvod, proč společnost Intel opustila číslované řady byl v tom, že jiné firmy, jako AMD, nebo Cyrix vyráběly levnější procesory kompatibilní s procesory Intel. A sekvence čísel nemůže být ochranou známkou. Po uvedení Pentií začalo AMD označovat svoje procesory jako K-5 (i586), K-6 (i586), K-7/Athlon (i686).

Prvním 32-bitovým procesorem byl 386, od něhož se procesory také označují jako IA-32. Společnost AMD zavedla x86_64, což je 64b verze. Později se k ní připojil i Intel, který má navíc IA-64, která ale není zpětně kompatibilní s IA-32.

Automounter

Automounter je démon, který hlídá přístup k adresářům, na kterých jsou odpojitelná zařízení. Když zjistí, že někdo chce s adresářem pracovat, připojí automaticky k adresáři zařízení. Můžete připojit cokoliv, co lze připojit příkazem `mount`. Obsah adresáře s přípojnými body automounteru by neměl být změněn, o mazání přípojných bodů ani nemluvě ;-).

Co potřebujete k použití automounteru? Podporu automounteru v jádře (parametr CONFIG_AUTOFS_FS nebo novější CONFIG_AUTOFS4_FS). Pokud si ho necháte zkompilovat jako modul, bude se jmenovat autofs resp. autofs4. Dále balík autofs (ve většině distribucí je už obsažen).

Autofs připojuje zařízení podle map. Pro každý adresář s body připojení je potřeba jedna mapa. Ta definuje jména adresářů sloužících jako přípojné body, typ souborového systému a volby pro příkaz `mount`.

Hlavním konfiguračním souborem je `/etc/auto.master`. Syntaxe je jednoduchá:

1. adresář s body připojení
2. soubory s mapami
3. parametry pro démona spravujícího přípojný bod

Dejme tomu, že chcete automaticky připojit jednotku cdrom a disketovou jednotku a chceme je mít v adresáři `/media`. Jejich mapu umístíme třeba do souboru `/etc/auto.media`. Do hlavního konfiguračního souboru pak přidáme řádek:

```
/media    /etc/auto.media    -g --timeout 60
```

Poslední část jsou parametry pro automount. `--timeout` určuje dobu nečinnosti v sekundách, po které se adresář automaticky odpojí. Asi by vám vadilo, že adresáře existují, až po jejich připojení. Když použijete parametr `-g`, vytvoří se "duchové" - adresáře budou existovat, i když na nich nebude nic připojeno.

Mapa zařízení určuje názvy přípojných bodů a jejich nastavení. Formát souboru je:

1. klíč (název přípojného bodu)
2. - parametry pro příkaz mount
3. : fyzické umístění

Do souboru `/etc/auto.media` můžeme přidat:

```
cdrom   -fstype=iso9660,ro                        :/dev/cdrom
disketa -fstype=auto,umask=000, \
         iocharset=iso8859-2,codepage=852          :/dev/fd0
winNT_C -fstype=smbfs,login=your_id,passwd=xxxxxx  WinNT:/C
zaloha  -fstype=nfs,soft                          mach1:/Backup
kernel  -ro,soft,intr              ftp.kernel.org:/pub/linux
cdcka   -fstype=autofs                      file:/etc/auto.cd
```

První položka je název přípojného bodu. Za ní je pomlčka, za kterou následují parametry příkazu `mount` oddělené čárkami. Přehled těch nejdůležitějších:

- `ro` - připojí pouze pro čtení
- `rw` - připojí pro čtení a zápis
- `blocksize` - velikost bloku zařízení
- `rsize` - velikost čtecího bufferu
- `wsize` - velikost zapisovacího bufferu
- `intr` - povolí zrušit připojování z klávesnice (pokud třeba systém čeká na server, který je odpojen)
- `nointr` - nepovolí zrušení připojování
- `soft` - když server neodpovídá po čase nastaveném parametrem `timeout` vrátí chybu a nebude se snažit o další spojení
- `hard` - zkoušet připojení stále, i když server neodpovídá; raději nepoužívejte bez nastaveného `intr`
- `bg` - pokoušet se o obnovu spojení na pozadí
- `fg` - pokoušet se o obnovu spojení na popředí
- `nosuid` - nepovolit používání SUID bitu na připojeném systému
- `suid` - opak předchozího
- `fstype=typ_ss` - typ souborového systému
- `async` - používat asynchronní režim pro čtení/zápis (používá se mezipaměť; zapsání změn se vynutí příkazem `sync`)

- `sync` - opak předchozího
- `auto` - připojit systém při provedení `mount -a`
- `noauto` - opak předchozího
- `user` - povolit připojení a odpojení běžnému uživateli
- `nouser` - povolit připojení a odpojení rootovi
- `timeout=cas` - čas, po kterém se přeruší pokusy o navázání spojení

Třetí položka je fyzická adresa zařízení. Pokud se jedná o váš počítač, není potřeba nic uvádět. Za dvojtečkou pak následuje jméno zařízení. Když je připojovaný souborový systém na síti, uvede se jméno počítače a dvojtečkou oddělená cesta k adresáři, který chceme připojit.

Nyní stačí restartovat démona startovacím skriptem, například `/etc/init.d/autofs restart` nebo `/etc/rc.d/init.d/autofs restart`.

Teď si můžeme vyzkoušet, jak nám automounter funguje. Zkuste vložit CD a zadat příkaz `ls -l /media/cdrom`. Jednotka se připojí a objeví se výpis adresářů na CD.

Můžete vytvořit i "podbdody" připojení. Například máte dvě jednotky CD a chcete je mít přístupné pod bodem připojení `/media/cdcka` a v adresářích `0` a `1`. To provede poslední řádek v ukázkové konfiguraci:

```
cdcka -fstype=autofs   file:/etc/auto.cd
```

Typ souborového systému je tu nastaven na autofs (souborový systém spravovaný automounterem) a poslední parametr je odkaz na mapu zařízení. Do souboru `/etc/auto.cd` pak vložíme:

```
0       -fstype=iso9660,ro      :/dev/hdc
1       -fstype=iso9660,ro      :/dev/hdd
```

Startovací skript přijímá kromě parametrů `start`, `stop` a `restart` ještě:

- `reload` - načte změny v konfiguraci
- `getmounts` - ukáže body připojení spravované automounterem
- `status` - ukáže nastavené a aktivní body připojení

Na jednotlivé body připojení samozřejmě můžete nastavit i symbolické odkazy. A nyní můžete vesele automountovat.

Na co se často ptáme: /etc/fstab

Soubor `/etc/fstab` je, jak už to tak v systémech unixového typu bývá, obyčejný textový soubor. Jeho účelem je systému popsat jednotlivé diskové svazky a vysvětlit mu, co, jak, kdy a podobně.

Fstab je na první pohled podivné slovo, ale je to jednoduchá zkratka ze slov **f**iles**s**ystem **tab**le, což můžeme volně přeložit jako "tabulka souborových systémů".

Jako do všech důležitých souborů má do něj přístup pouze velitel systému (chcete-li, třeba root). Ten jako jediný může rozhodovat o osudu jednotlivých svazků.

Těmito svazky jsou obvykle jednotlivé oddíly disků, ale není to pravidlem. Fstab může vlastně popisovat cokoliv, co lze připojit příkazem mount. Malý příklad: Chceme připojit disketu do adresáře `/mnt/floppy`. První disketová jednotka je označená v systému jako fd0, takže jako root zadáme příkaz

```
# mount /dev/fd0 /mnt/floppy -t vfat
```

Tímto příkazem říkáme, že chceme připojit blokové zařízení `/dev/fd0` do adresáře

`/mnt/floppy` a že je na něm souborový systém typu vfat.

Představte si, že byste takový příkaz vypisovali při každém připojení jednotlivého média. Navíc byste znemožnili práci s disky obyčejným uživatelům, kteří nemají práva na připojování a odpojování svazků.

Už je vám asi jasné, že právě toto řeší soubor `fstab`. Jeho syntaxe je prostá: co řádek, to svazek. Na každém řádku je několik položek oddělených standardními oddělovači (mezerami nebo tabulátory). Pořadí je podstatné a nezaměnitelné a musíme jej dodržet.

Malý příklad takového řádku:

```
    1          2           3         4          5 6
/dev/fd0  /mnt/floppy   vfat   noauto,user   0 0
```

1. zařízení, které budeme připojovat.
2. připojovací bod (adresář), do kterého se svazek připojí
3. typ souborového systému
4. parametry pro připojení
5. určuje, jestli bude svazek zálohován
6. nastavuje, jako kolikátý bude svazek kontrolován při startu

Tohle je stručné vysvětlení jednotlivých položek. Většina z nich si zaslouží podrobnější popis, takže jdeme na to:

První položka
: Může ji tvořit jak běžný oddíl disku (`/dev/hda2`), tak i označení vzdáleného svazku, třeba přes SMB (`//kuchyn/dokumenty`).

Druhá položka
: Adresář připojení. Obvykle je to některý z podadresářů `/mnt`. Pro swap se zadává `none`, protože ten se nikam nepřipojuje.

Třetí položka
: Kompletní seznam souborových systémů, které podporuje vaše jádro naleznete v `/proc/filesystems`. Jejich množství a typ záleží na verzi jádra a zavedených modulech. Mohou to být: proc, tmpfs, ext2, ramfs, iso9660, devpts, ext3, usbdevfs, usbfs a další. Můžete také použít výraz auto a jádro se pokusí obsah samo detekovat.

Čtvrtá položka
: Do té je možno vyplnit celou řadu různých parametrů pro různé souborové systémy. Parametry se od sebe oddělují čárkou. Mezi hlavní patří
  - `noauto` (nepřipojuje svazek automaticky při startu)
  - `users` (s tímto svazkem mohou pracovat i běžní uživatelé)
  - `codepage` (znaková sada, ve které jsou názvy souborů)
  - `iocharset` (znaková sada, do které se budou převádět názvy souborů)
  - `noexec` (nespouštěj soubory na tomto médiu)
  - `umask` (nastavení práv u souborů)
  - `ro` (read only - pouze ke čtení)
  - `rw` (read write - čtení i zápis)

  Kompletní popis naleznete v manuálové stránce k mount.

Pátá položka
: Tuto položku používá program `dump`, který podle ní pozná, zda má provádět zálohu svazku. Jednička znamená zálohovat, nula nezálohovat.

Šestá položka

Poslední položku používá program `fsck` aby zjistil, jako kolikátý bude svazek kontrolován. Jednička by měla označovat kořenový svazek (připojuje se do `/`), dvojka pak ostatní a nula ty, které se kontrolovat nebudou.

Každý řádek (tedy i poslední) musí být ukončen enterem (odřádkován). Podle něj pak programy poznají, že informace o svazku končí. Častou chybou je nezapsání enteru na konec posledního řádku. Pokus o připojení pak končí chybou.

Oddíly označené jako swap, jsou připojovány při startu systému pomocé příkazu `swapon -a` Stejně tak je lze odpojit pomocí `swapoff -a`

Soubor jsme si popsali a teď se ještě podíváme na nejčastěji kladené otázky, čili FAQ:

Mám divná práva na FAT. Co s tím?

Souborový systém FAT (a některé další) neumožňují ukládání práv. Proto souborům na nich jádro přidělí jen virtuální práva. Toto chování lze nastavit pomocí parametrů `uid` a `gid`, za nimiž následují id čísla uživatele a skupiny, které budou patřit soubory na daném svazku. Dalším důležitým parametrem je `umask`, který nastavuje bity, které ve výsledku **nebudou** u souboru nastaveny. Nejčastěji chceme, aby nebyly všechny soubory spustitelné. To nám zajistí právě umask. Chceme vypnout spouštěcí právo, což je první bit. Problém ale nastane s adresáři, které přijdou o právo x a nebude možno do nich vstoupit. To řeší parametr `dmask`, který pracuje stejně, ale týká se pouze adresářů. Příklad:

```
/dev/hda3  /mnt/fat  vfat umask=111,dmask=000,gid=500,uid=500 0 0
```

Mám rozhozenou češtinu. Dá se to nějak opravit?

Při dotazu na souborový systém dostane jádro názvy souborů v určitém kódování. Tak se vám i zobrazí. Součástí jádra je ale i mechanismus pro jejich převedení do kódování, které používáte. K tomu slouží parametry codepage a iocharset. Příklad převedení kódování z 852 do iso8859-2:

```
/dev/hda3  /mnt/fat  vfat iocharset=iso8859-2,codepage=852 0 0
```

Jak se zbavit chybové hlášky při kopírování na FAT?

Díky tomu, že FAT nemá zmíněná práva, nedaří se jádru do něj tyto informace dostat. K tomu, abychom nebyli pořád upozorňováni, souží parametr `quiet`. Příklad:

```
/dev/hda3  /mnt/fat  vfat  quiet  0 0
```

Proč můžu připojit médium jen jako root?

Tak je to v unixu zařízeno. Standardně je možno se zařízeními pracovat jen jako root. Asi byste nebyli rádi, kdyby vám nějaký uživatel na serveru odpojil půlku věcí. Pokud ale chcete i uživatelům dovolit připojovat/odpojovat některé svazky (obvykle cdrom, diskety, apod.), použijte parametr users. Přiklad:

```
/dev/cdrom  /mnt/cdrom  iso9660  users 0 0
```

Co znamená v mém nastavení slovo supermount?

Některé "přítulné" distribuce používají k automatickému připojování démona, který sám sleduje, co se děje a automaticky připojuje/odpojuje zařízení. fstab používá stále jako svůj konfigurační soubor. Řádek pro supermount může vypadat třeba takto:

```
none /mnt/cdrom supermount dev=/dev/hdc,fs=auto,ro,--
,iocharset=iso8859-1,codepage=850 0 0
```

Položky jsou velmi podobné, jen místo souborového systému je potřeba zapsat supermount a do parametrů přidat několik informací pro démona.

Některé řádky v mém fstab mají zvláštní cesty a parametry. K čemu tam jsou?

Jádro podporuje některé speciální souborové systémy jako tmpfs a procfs. Oba jsou to virtuální souborové systémy. Tmpfs slouží jako ramdisk. Data, která na něj uložíme, zůstávají v paměti a neukládají se nikam na disk. Procfs je standardně připojen v adresáři `/proc` a umožňuje získávat důležité informace od jádra. Opět doopravdy neexistuje a jádro jej vytváří virtuálně.

Jak pomocí fstabu připojím NFS?

NFS je standardní unixový způsob sdílení dat po síti. Máte-li přístup k síťovému svazku na NFS, stačí přidat řádek, jehož prvním parametrem bude název serveru a celá cesta ke svazku:

```
server:/usr/local/pub  /mnt/pub  nfs  ro  0 0
```

Nfs má některé svá specifika a řadu parametrů, kterými lze ovlivnit jeho chování. Doporučuji proto prostudovat manuálovou stránku nfs(5).

/etc/fstab

Fstab je jednoduchá zkratka ze slov **f**ile**s**ystem **tab**le, což můžeme volně přeložit jako "tabulka souborových systémů".

Soubor /etc/fstab je obyčejný textový soubor, jehož účelem je systému popsat jednotlivé diskové svazky a vysvětlit mu kam a jakým způsobem je má připojovat.

Těmito svazky jsou obvykle jednotlivé oddíly disků, ale není to pravidlem. Fstab může vlastně popisovat cokoliv, co lze připojit příkazem mount.

Více informací o problematice fstab naleznete v podrobném článku Na co se často ptáme: /etc/fstab.

portage

Nejznámějším a nejzákladnějším příkazem z Portage je emerge.

Systém správy balíčků distribuce Gentoo. Seznam balíčků je uložen v adresářové struktuře dělené podle kategorií v `/usr/portage`.

Články na abclinuxu.cz:
- Balíčkovací systém Gentoo Linuxu - I
- Balíčkovací systém Gentoo Linuxu - II
- Gentoo Linux 1.4

distribuce

Je to kompletní operační systém (nejčastěji GNU/Linux), včetně aplikací a konfiguračních nástrojů. Různé distribuce jsou zaměřeny na různé oblasti použití - na servery, pro malá zařízení, pro běžné uživatele, ... Proto jsou mezi nimi docela velké rozdíly.

Články na abclinuxu.cz:
- Co je to Linux
- Téma: výběr distribuce

Linux

1. jádro (základní část) operačního systému pod licencí GNU GPL, který začal vyvíjet Linus Torvalds.
2. Obecné označení distribuce operačního systému se stejnojmenným jádrem. Někdy se označuje jako GNU/Linux.

Linus napsal **právě a pouze to jádro**, které se později jako poslední chybějící člen začlenilo do operačního systému GNU, který má na svědomí Free Software Foundation (FSF) založená Richardem Matthewem Stallmanem (RMS). Dnešní Linuxy se ale od myšlenky GNU už dost vzdalují a jako potomek GNU může být označen asi jen Debian GNU/Linux. Označení Linux se dnes používá především ve zobecněném významu a myslí se jím celý operační systém (některá z distribucí), nikoliv jen jádro.

Články na abclinuxu.cz:
- Co je to Linux

Gentoo

Source based distribuce GNU/Linuxu, která používá vynikající systém správy balíčků jménem portage.

Články na abclinuxu.cz:
- Balíčkovací systém Gentoo Linuxu - I
- Balíčkovací systém Gentoo Linuxu - II
- Gentoo Linux 1.4

emerge

Program `emerge` je součástí Portage, což je balíčkovací systém Linuxové distribuce Gentoo. Je napsaný, jako celá Portage, v Pythonu. Skvěle jsou vyřešeny závislosti, ale bohužel s přibývajícími ebuildy začíná být dost pomalý. Naneštěstí je kód slušně nepřehledný, jinak už by se do zrychlení jistě spousta vývojářů pustila, přestože jde hlavně o to, udělat jakousi databázi, díky které bude prohledávání ebuildů daleko rychlejší, jenže to má určitě spoustu háčků...

glibc

Glibc je GNU C knihovna, což je jedna ze základních součástí systému, v embeded zařízeních se nahrazuje knihovnou uclibc. Tato knihovna je distribuována pod licencí LGPL a je využívána téměř všemy programy psanými v jazyce C.

Na co se často ptáme: MPlayer

Nejrůznější problémy s MPlayerem jsou vděčným tématem mnoha otázek v diskuzi na AbcLinuxu.cz. Pokusím se tedy o co nejpodrobnější popis.

[V článku zmiňované softwarové balíky mohou mít v různých distribucích odlišná pojmenování. Názvy, které uvádím, vycházejí z Debianu, ale nepředpokládám, že by se měly od jiných distribucí lišit nějak zásadně. Každopádně vždy připojím ještě odkaz na stránky daného projektu.]

Parametry skriptu `./configure`

Ačkoliv lze pro mnoho distribucí sehnat předkompilované balíčky, nebudu se jim zde z pochopitelných důvodů věnovat. Nemohu sice soudit, nakolik vlastní kompilace zaručí vyšší výkon, ale vím zcela jistě, že v binárním balíku bude zakompilováno mnoho věcí, o které vůbec nemusíte stát - a naopak jiné mohou chybět. Jejich jedinou výraznou výhodou proto zůstává skutečnost, že budete ušetřeni instalace -dev (-devel) verzí knihoven, jejichž podporu chcete v programu mít. Každopádně počítejte s knihovnami XFree (xlibs).

Kompilace však není nic složitého. Může se stát, že narazíte na problém s použitou verzí kompilátoru. Dokumentace nedoporučuje gcc verze 2.96 a 3.0.x. Ale najde se dost lidí (včetně mě), kteří různé verze s těmito kompilátory zkompilovali bez problémů. Chyba se však někdy může projevit až po 'úspěšné' kompilaci - např. okamžitým pádem - ještě než se program pořádně spustí... Ačkoliv tedy nemohu poskytnout stoprocentní radu, alespoň dodám, že zatím jsem ještě nikdy nenarazil na problém s verzí gcc, která byla totožná s verzí, kterou byl kompilován kernel (viz `cat /proc/version`).

Většinu parametrů není potřeba vůbec zadávat. `./configure` skript se pokusí automaticky rozpoznat přítomnost všech možných zařízení, ovladačů a knihoven, které máte v systému nainstalovány. Je proto důležité mít je na standardních místech, aby je skript nalezl - ale to vlastně platí pro každou kompilaci.

Začněte stažením všech externích (windowsích) kodeků, které budete chtít používat (stabilní vydání v sobě obsahují balík libavcodec, takže [úctyhodná] základní funkčnost je zajištěna i bez dalších kodeků). Protože by byla škoda ochudit se o bohaté možnosti MPlayeru, doporučuji nainstalovat všechny (nebo alespoň Quicktime) - alespoň vás později žádný formát nezaskočí. Program bude tyto kodeky standardně hledat v adresáři `/usr/lib/win32`. Pokud vám toto umístění z nějakého důvodu nevyhovuje, můžete `./configure` skript nasměrovat jinam pomocí parametru `--with-win32libdir`.

Nezapomeňte, že kodeky je potřeba "nainstalovat" (resp. nakopírovat) ještě PŘED spuštěním skriptu `./configure` a kompilací - jinak pro ně nebude zakompilována podpora.

Pokračujeme stažením poslední verze (v současné době 1.0pre1) samotného MPlayeru. Archív rozbalte a přepněte se do vzniklého adresáře. Třeba takto:

```
tar -jxvf MPlayer-číslo.verze.tar.bz2
cd MPlayer-číslo.verze
```

Nastal čas ke spuštění skriptu `./configure`. Abychom však dosáhli kýženého výsledku, je ještě nutné se přesvědčit, jestli jsou v systému nainstalované a k dispozici všechny potřebné vývojové [development, -devel, -dev] knihovny. Můžeme si je pro přehlednost pracovně rozdělit do čtyř skupin:

1. Podpora různých formátů obrazu nebo zvuku.
2. Výstup grafické karty.
3. Výstup zvukové karty.
4. Různé funkce přehrávače.

1. Audio a video formáty

V této oblasti jsme se již o to nejdůležitější postarali instalací win kodeků. Pokud požadujete i podporu obrázkových formátů (PNG, JPEG, GIF), nainstalujte -dev balíky knihoven libpng, libjpeg, giflib.

Aby bylo možno pracovat s formátem MP3 (vytvářet), nainstalujte liblame. Pro OGG budete potřebovat libogg/libvorbis.

2. Výstup grafické karty

Zmíním se o nejčastěji používaných kartách. Všechny budou fungovat i bez jakékoliv zvláštní přípravy, ale pak nebudete moci využít hardwarové akcelerace, což se podepíše na výkonu - především při celoobrazovkovém módu.

Matrox (G200/G400/G450/G550)

MPlayer nabízí jaderný modul mga_vid. Můžete ho používat jak ve framebufferové konzoli, tak v XWindow.

Nejprve si jej však musíte zkompilovat, vytvořit zařízení `/dev/mga_vid` a modul natáhnout do kernelu. Skript `./configure` toto zařízení nalezne a podporu automaticky zapne. Z hlavního adresáře zdrojáků MPlayeru zadejte:

```
cd drivers
make
mknod /dev/mga_vid c 178 0
insmod mga_vid.o
```

Abyste měli `/dev/mga_vid` připravené i po rebootu, zkopírujte zkompilovaný modul `mga_vid.o` do adresáře `/lib/modules/verze_kernelu/někam`, přidejte do vašeho `/etc/modules.conf` souboru řádek:

```
alias char-major-178 mga_vid
```

a aktualizujte seznam natahovaných modulů - většinou `depmod -a`.

V Debianu vložte řádek s "alias..." do samostatného souboru `mga_vid`, který umístěte do `/etc/modutils/`. Nakonec spusťte `update-modules`.

Pokud používáte DevFS, můžete vynechat příkaz "mknod...". mga_vid je už nějakou chvíli DevFS kompatibilní a soubor zařízení `/dev/mga_vid` se po natažení modulu vytvoří automaticky. Abyste se opět dočkali zařízení i po rebootu, vložte do vašeho souboru `devices` pro DevFS řádek (v Debianu samostatný soubor s takovým řádkem do `/etc/devfs/devices.d/`):

```
mga_vid c 178 0 root video 0660
```

3Dfx

Máte dvě možnosti. Buď použít novější a kvalitnější ovladač `tdfx_vid`, který se instaluje na vlas stejně jako předchozí `mga_vid` -- a nebo využít standardní jaderný ovladač `tdfxfb.o`. Většina distribučních jader by ho měla mít zkompilovaný. Pokud máte vlastní jádro, přikompilujte si ho. `./configure` skript budete ještě muset instruovat parametrem `--enable-tdfxfb`.

ATI

K dispozici jsou VIDIX ovladače. MPlayer by měl vaši kartu správně rozpoznat a automaticky použít ten správný (mach64_vid, rage128_vid a radeon_vid). Pak je tu ještě možnost použít overlay ovladač `radeon_vid`, který můžete najít a zkompilovat v adresáři `drivers/radeon`. (Postup opět stejný jako v předchozích dvou případech - s tím rozdílem, že ovladač potřebuje ke své funkci VESA BIOS, což se projeví na podobě parametru pro výběr video výstupu při spouštění přehrávače. Více viz `drivers/radeon-README`

S3

Standardní xv ovladač by měl ve většině případů posloužit optimálně.

nVidia

Použijte ovladače přímo od fy. nVidia. Pokud máte novější kartu, bude xv ovladač fungovat bez problému. Riva128 bohužel Xvideo rozšíření nepodporuje (resp. firemní ovladač ne).

Kromě ovladačů specifických pro konkrétní typy grafických karet jsou k dispozici ještě zvláštní možnosti grafického výstupu, které vám usnadní použití MPlayeru v nejrůznějších podmínkách. Pokud budete mít o zakompilování podpory zájem, opět nainstalujte příslušné knihovny.

- **SDL (libsdl - Simple DirectMedia Layer)**: Pokud vás zvláštní okolnosti nenutí využívat tento výstup (i audio), oceníte ho asi jen kvůli tomu, že umožňuje zobrazovat titulky k filmu mimo vlastní obraz (tedy pouze relevantní u širokoúhlých záznamů).
- **Framebuffer na konzoli** fbdev (vyžaduje zakompilovanou podporu pro vaší kartu v kernelu): Když už používáte konzoli ve vysokém rozlišení a frekvenci, proč si v ní ještě nepustit film?
- **SVGALib**: Další z možností, jak si v konzoli pustit video. V tomto případě je to záležitost instalace a konfigurace jediné knihovny.
- **AAlib**: Skutečně velmi kvalitní způsob zobrazování grafiky pomocí běžných ASCII znaků. Sice ne moc použitelné, ale určitě zajímavé.

3. Výstup zvukové karty

Nejčastěji používanými ovladači jsou buď jaderné OSS nebo od projektu ALSA. MPlayer je na konfiguraci zvukového výstupu dost citlivý a může vám klidně oznámit, že váš počítač není dost výkonný, aby mohl soubor XY přehrát. Pokud skutečně nesedíte u pomalého stroje, bude důvodem nejpravděpodobněji nevyhovující OSS ovladač zvukové karty. Budete přinuceni vyzkoušet ALSA ovladače, což však není na škodu, protože jsou obecně kvalitnější a podporují více funkcí zvukových karet.

Ovladače správně nainstalované v systému skript opět nalezne a přidá k nim i podporu aRts a esd, pokud k těmto systémům máte nainstalované potřebné vývojové knihovny. Jejich zakompilováním pak získáte možnost nasměrovat audio výstup z MPlayeru na vámi používaný zvukový server.

4. Různé funkce přehrávače

Několik dalších užitečných parametrů:

- `--language=cz` Aby na vás program mluvil česky. Nevýhodou je, že případná chybová hlášení jsou někdy také česky, což prakticky znemožňuje dohledání jejich významů na internetu.
- `--enable-gui` Zakompiluje podporu grafického rozhraní. Budete potřebovat knihovny Gtk+ 1.2 (libgtk).
- `--enable-menu` Zapne podporu obrazovkového menu. Sice poměrně zbytečné, ale hezky to vypadá
- `--enable-dvdnav` Pokud chcete přehrávat DVD včetně jejich menu a rádi byste vyzkoušeli experimentální podporu knihovny libdvdnav, použijte tento parametr. Moc si toho však od něj neslibujte, protože moc nefunguje.
- `--disable-runtime-cpudetection` Pokud budete mít autodetekci procesoru zapnutou, upozorní vás MPlayer při každém spuštění, že to není optimální. Ačkoliv podle nápovědy ke skriptu je tato funkce ve výchozím nastavení vypnuta, vždy se mi při kompilaci zapnula. Vyřešil to až tento parametr.
- `--enable-xinerama` Zakompilovat podporu pro zobrazení na více obrazovkách.

Kompilace

Až budete spokojeni s vybranými parametry kompilace, spusťte `./configure` skript a nezapomeňte zkontrolovat jeho výstup. Na úplném konci naleznete šikovný přehled zapnutých a vypnutých parametrů v jednotlivých oblastech.

Takto vypadá příkazová řádka většinou když kompiluji MPlayer:

```
./configure --disable-runtime-cpudetection --enable-dvdnav --
enable-menu --enable-gui --language=cz
```

A takhle vypadá závěrečné shrnutí. Pokud najdete některou z požadovaných funkcí/knihoven/ovladačů v "Disabled" části, ačkoliv byste ji radší viděli v "Enabled", opravte instalaci příslušných knihoven ve vašem systému a pak spusťte `./configure` (se všemi parametry) znovu.

```
Enabled optional drivers: [zapnuté/nalezené funkce]
  Input: streaming edl tv mpdvdkit2 vcd
  Codecs: qtx divx5linux xvid libavcodec dshow/dmo win32 libvorbis
libmad
  Audio output: alsa9 arts oss nas sdl mpegpes(file)
  Video output: xvidix sdl vesa jpeg png mpegpes(file) svga aa xmga mga
opengl dga xv x11
Disabled optional drivers: [vypnuté/nenalezené funkce]
  Input: tv-v4l tv-bsdbt848 cdda dvdnav dvdread dvdcss smb
  Codecs: divx4linux libdv real xanim liblzo gif
  Audio output: sgi sun esd dxr2 win32
  Video output: bl zr dxr3 dxr2 directx gif89a fbdev ggi directfb
tdfxfb 3dfx
```

Zkompilujte příkazem `make` a proběhne-li vše bez nějakého zádrhelu, nainstalujte program: `make install` (jako root, třeba pomocí `su`). Později můžete MPlayer odstranit spuštěním příkazu `make uninstall` ze stejného adresáře. Pro Debian si můžete snadno vytvořit .deb balíček. Informace v souboru `debian/README.debian`.

Konfigurace

Ačkoliv je možné veškerou konfiguraci provádět z příkazové řádky, pohodlnější je nastavit základní parametry v konfiguračních souborech. Pokud vás to neláká, máte možnost použít konfigurační nástroj, který je součástí GUI. Hodnoty nastavené v něm sice budou platit pouze pro GUI verzi programu, ale pokud rozhraní příkazové řádky stejně nehodláte používat, může vám to být jedno. Nastavení v tomto Gtk+ prostředí je celé počeštěno a navíc velmi přehledné. Na tom skutečně není co vysvětlovat. Zmíním tedy jen doinstalaci samotného GUI (skinu) -- přestože jsme jeho podporu zakompilovali, žádné nainstalováno nebylo; MPlayer ve svém základním "balení" žádné neobsahuje.

GUI

Stáhněte některý ze skinů, rozbalte a výsledný adresář přesuňte buď do `/usr/local/share/mplayer/Skin/` (pokud jste ovšem `./configure` skriptu nezadali nějaký jiný instalační adresář v parametru `--prefix`) nebo do `~/.mplayer/Skin`. První varianta je samozřejmě celosystémová, druhá pouze pro právě přihlášeného uživatele. Adresář se skinem pak přejmenujte na `default` (pokud chcete skiny obměňovat, bude výhodnější vytvořit pouze symbolický odkaz: `ln -s` *`adresar_se_skinem`* `default`). MPlayer v GUI podobě se spouští příkazem `gmplayer`.

Konfigurační soubory

Hlavní celosystémový konfigurační soubor umístěte do `/usr/local/etc/mplayer.conf`, jeho uživatelský protějšek (který má při běhu programu přednost) pak do `~/.mplayer/config`. Jako šablona vám může posloužit soubor `etc/example.conf` ze zdrojového adresáře.

`example.conf` je velmi bohatě komentovaný, takže nastavení by nemělo činit zvláštní potíže. V následující tabulce jsou proto vyjmenovány jen ty nejdůležitější parametry (parametry pro příkazovou řádku, které jsou vysvětlovány na jiných místech článku, můžete také zařadit do konfiguračního souboru - jen zápis se změní např. takto: `-subcp cp1250` -> `subcp=cp1250`).

Parametr	Význam
`vo`	video výstup; spusťte `mplayer -vo help`, což vypíše dostupné ovladače.
`ao`	audio výstup; `mplayer -ao help` opět vypíše tentokrát audio ovladače. Nezapomeňte, že pokud již máte spuštěný nějaký program obsluhující systémové ovladače (ALSA, OSS) zvukové karty, budete muset jako výstup pro MPlayer použít právě ten (tzn. např. aRts v KDE).
`gui = yes`	Vždy používej GUI řežim.
`skin =` *`nazev_adresare`*	Použij skin ze zadaného adresáře.
`framedrop = yes`	Když nestíhá ovladač/procesor/atd., vynechávej snímky.

Fonty

Fonty pro titulky k filmům a pro zobrazení obrazovkového menu. Máte několik možností, jak se s touto věcí vypořádat. Asi nejjednodušší je použít relativně novou funkčnost MPlayeru - totiž schopnost používat jakýkoliv truetype font tak, jak je. Podmínkou je mít při kompilaci správně nainstalované vývojové knihovny FreeType2. Pak už je jen potřeba opatřit si pěkné (české) fonty, což můžete řešit třeba instalací sady microsoftích fontů pro web. Jednotlivé fonty si v akci můžete vyzkoušet při spuštění s parametrem `-font` *`/cesta/k/fontu/nazev_fontu.ttf`* (více o titulcích v části "Praxe"). Od požadovaného fontu pak vytvořte symlink do adresáře MPlayeru ve vašem home:

`ln -s /cesta/k/fontu/`*`nazev_fontu`*`.ttf ~/.mplayer/subfont.ttf`

Pokud vám uvedený postup z nějakého důvodu nevyhovuje (například nechcete instalovat knihovny FreeType - i když nechápu, proč by se mohl někdo chtít o její skvělé funkce ochuzovat...), můžete využít připravené baličky s fonty, které můžete získat přímo na stránkách MPlayeru. Obsah takového balíčku pak nakopírujte do adresáře `~/.mplayer/font/`.

Kódování češtiny

Menší - ale důležitá - vsuvka. Nebudu se pouštět do žádných obšírných vysvětlení. Chcete-li si přečíst něco více o tom, proč vůbec máme nějaká kódování a kde se vzala, zkuste třeba stránky cestina.cz. Při používání MPlayeru budete asi nejčastěji hledat řešení pro přehrávání filmů s externími titulky, které jsou ve většině případů v kódování Win1250. Pokud se vám nechce titulky pomocí některého z mnoha konverzních programů (nebo online) převést do kódování, které používá vaše instalace Linuxu (tj. ISO8859-2, případně UTF-8), budete muset MPlayeru říct, jaké kódování má v textovém souboru titulků očekávat.

K tomu slouží parametr `-subcp` *`kódování`*. Pokud používáte pro zobrazení titulků rovnou truetype fonty (viz výše), máte po starostech. Usnadnit život vám ještě může utilitka Enca, kterou napsal nestor diskuzního fóra AbcLinuxu, David Nečas (Yeti). Enca je velmi šikovná v rozpoznávání jednotlivých kódování, takže pokud si ji nainstalujete (zkompilovanou s podporou programu iconv, jehož označení různých kódování MPlayer používá), můžete parametru `-subcp` naservírovat "hodnotu" `` `enca -i `*`soubor_s_titulky.sub`*`` `. Nemusíte si pak dělat hlavu s tím, v jakém kódování titulky vlastně jsou.

Při použití balíčků s fonty je situace trošku složitější. Fonty jsou totiž většinou určeny jen pro jedno kódování (ty, na které ukazuje odkaz výše, jsou pouze pro kódování ISO8859-2). Můžete zkoušet různé kejkle popisované na spoustě míst, ale řekl bych, že úplně nejprostší řešení je zároveň to

nejlepší. Nahradíte-li soubor `font.desc` z balíčku fontů takovým, který je upraven i pro zobrazení českých znaků definovaných v kódování Win1250, máte opět po starostech. Stáhněte si jej třeba odsud (upravil Artwine).

Praxe

Předpokládám, že nejnázornější bude ukázat vše na příkladech.

Běžné použití

Chcete-li přehrát jakýkoliv soubor bez dalších speciálních nastavení (a za předpokladu, že v konfiguračním souboru už máte nastavené preferované výstupy pro video i audio), spusťte MPlayer s názvem (a případně cestou k němu) tohoto souboru jako jediným parametrem:

```
mplayer film.avi
```

Pokud chcete již popisované nastavení z konfiguračního souboru "přebít", vložte parametr na příkazovou řádku. Třeba:

```
mplayer -ao alsa9 http://server.cz/radio.pls
```

DVD, VCD, SVCD, CD

Přehrávání všech těchto disků je velmi snadné, a pokud jej budete chtít provozovat častěji, doporučuji vytvořit pro jednotlivá zařízení symbolické linky, abyste nemuseli MPlayeru vždy na příkazové řádce říkat, kde to DVD/CD hledat. DVD mechaniku očekává program na `/dev/dvd`. CD zase na `/dev/cdrom`.

```
mplayer -dvd 1 -dvd-device /dev/hdc
```

Kde `-dvd 1` značí stopu na DVD. Uvedený příkaz by toho ve většině případů moc nepřehrál, protože stopa 1 bývá obyčejně několikavteřinové logo produkční společnosti nebo copyrightové upozornění.

Na tomto místě stojí za zmínku nastavení DMA vaší CD/DVD mechaniky. Při použití staršího PIO režimu se budete potýkat s pomalým přenosem => přehráváním. Současný stav zjistíte třeba pomocí `hdparm -v /dev/dvd`. U konkrétní mechaniky pak můžete DMA zapnout parametrem `-d1`.

Titulky

V části, která se zabývala fonty, jsem už mluvil o kódování titulků. Dalším problémem však bývá, že MPlayer titulky u širokoúhlého filmu nezobrazí hezky v černém pruhu pod obrazem, nýbrž nešikovně do obrazu. Jsou dva způsoby, jak tomu předejít.

Prvním a systémovějším řešením je říct MPlayeru, kam chceme, aby titulky umístil, pomocí parametru `-vop`. Tento parametr se používá pro nastavení všech filtrů/pluginů, které MPlayer zná. Jejich seznam získáte příkazem `mplayer -vop help`, bližší popis pak v `man mplayer`. Nás teď zajímá filtr "expand", který zvětší videovýstup o zadané hodnoty (obraz zůstane nedotčen, nejedná se manipulaci s rozlišením filmu). Budeme předpokládat, že soubor s titulky se jmenuje `film.sub` a tím pádem ho MPlayer načte automaticky.

```
mplayer -vop expand=0:-100:0:0 film.avi
```

Ke spodnímu okraji obrazu jsme tímto přidali 100 bodový černý pruh, který je pro zobrazení titulků plně dostačující. Při přepnutí do celoobrazovkového režimu se toto nastavení projeví posunutím obrazu filmu o úměrný počet bodů výš.

Druhou možností je využití video výstupu SDL (`-vo sdl`), který titulky automaticky umístí do volného prostoru pod obrazem (tzn. pouze v celoobrazovkovém režimu).

Závěr

Tento článek nemůže a ani nechce být kompletním průvodcem. To už by bylo snazší přeložit celou dokumentaci. Přesto jsem se snažil nevynechat žádnou oblast, která obvykle (alespoň to tak vypadá podle otázek v diskuzním fóru) dělá uživatelům problémy. Opravy a doplňující informace jsou samozřejmě vítány.

Čím v Linuxu nahradit aplikace Windows?

Seznam je rozdělen podle druhu využití programů. Řazení jednotlivých programů je buď náhodné nebo založené na mé osobní preferenci. Máte-li pocit, že jsem opomněl zmínit program, který je podle vašeho názoru vhodným kandidátem na zařazení do tohoto přehledu, neopomeňte jej vy zmínit v diskuzi - bude zařazen přímo do tabulky.

Použil jsem čtyři druhy zkratek: **K**, **G**, **T** a **CZ**.

K: Program pro prostředí KDE. Ačkoliv jej lze spouštět bez problému (jsou-li v systému potřebné knihovny) i z jakéhokoliv jiného Window Manageru (nebo přímo v X Window), autor přizpůsobil vzhled uživatelského rozhraní, a potažmo tedy způsob ovládání, pro desktop KDE.

G: Program pro prostředí Gnome. Platí totéž, co u programů pro KDE - jen je to tentokrat Gnome.

T: Program pro textové prostředí - konzoli. V systému X Window pracuje v terminálovém emulátoru (xterm apod.).

CZ: Odkaz na přídavné počešťovací soubory. U mnoha programů je české prostředí k dispozici přímo v programu - buď se nastaví automaticky podle locales (např. Midnight Commander), nebo je třeba zvolit jej jako možnost při kompilaci (např. MPlayer). Většina aplikací pro KDE a Gnome je počeštěna po nainstalování lokalizačního balíčku. Ani v jednom z těchto případů pak v seznamu odkaz CZ není.

Zároveň se omlouvám všem, kteří by rádi viděli i zvláštní kategorie pro programy psané na míru třeba Window Makeru nebo Afterstepu. Pokud se takové aplikace v přehledu vyskytují, nechal jsem je splynout s "šedí" ostatních programů pro X.

Rozhodl jsem se do přehledu zařadit i programy, které jsou ve verzích pro Windows i Linux. Možná někdo zajásá, že oblíbený program vůbec měnit nemusí - stačí používat linuxovou verzi.

V některých případech nemusí být zařazení úplně ideální. Důvodem je častá univerzálnost programů.

Výčet programů pro Windows není samozřejmě vyčerpávající, ale pouze ilustrační.

Poskytnuté odkazy na stránky, kde lze stáhnout jednotlivé programy, nemusí být vždy nejlepším způsobem, jak program získat. Většina linuxových distribucí poskytuje pro své uživatele předpřipravené balíčky (viz odkazy), které lze do systému snadno doinstalovat, aniž by bylo třeba kompilovat zdrojový kód. Pokud je balíček dobře sestaven, umístí také jednotlivé soubory tam, kde je vaše distribuce očekává.

Internetové prohlížeče, browsery
Opera, Netscape, Mozilla, Firefox, Internet Explorer

- Opera (CZ), Netscape, Mozilla (CZ), Firefox (CZ)
- ... a ještě: Konqueror (K), Galeon (G)
- minimalistické (malé a rychlé): Dillo, Skipstone , Arachne
- zvláštní kategorie: Emacs/W3 :)
- a nakonec: Lynx (T) a Links (T), ELinks

Poštovní klienti
Outlook (Express), Eudora, Pegasus Mail, The Bat! + Mozilla Mail a Netscape Mail

- Evolution (G), Balsa (G), Cronos II (G), KMail (K) a KMail Cool (K), Aethera
- Netscape Mail, Mozilla Mail (CZ), Thunderbird (CZ)
- gtkmail, Pronto!, Liamail, Sylpheed
- Mutt (T), Pine (T)

Správce souborů
Windows Commander, Průzkumník, WinNC, Norton Commander, M602
Dvou (a více) panelové:

- Krusader (K), Gnome Commander (G),
- Seksi Commander, Tux Commander, PFM, BF-Commander, X Northern Captain, emelFM, gentoo, FileRunner, TkDesk, Xfe, Pfm
- Midnight Commander (T), deco (T)

Jako Průzkumník:

- Konqueror (K), Nautilus (G)

Kancelářské balíky
MS Office, 602Pro PC Suite, WordPerfect Office
Zdarma

- OpenOffice.org (CZ), KOffice (K), Gnome Office (G)

$$$

- Hancom Office, StarOffice 6.0, WordPerfect Office

Grafické programy
Paint Shop Pro, Adobe (Photoshop, Illustrator), Corel (PhotoPaint, DRAW)
Bitmapové:

- The Gimp, Hancom Painter ($$$), Panorama Tools

Vektorové:

- OpenOffice.org Draw (CZ), Skencil, Sodipodi (G), Figurine, Ipe, Inkscape

Raytracing, modeling:

- POV-Ray, Blender, Radiance, Giram, Innovation 3D, Behemot

Prohlížeče obrázků
IrfanView, ACDSee

- ShowImg (K), Gwenview (K), Kuickshow (K), Fotoon (K), KimDaba (K), GImageView (G), GQview, XnView
- ImageMagick (T, nejen) Pho (T), Feh (T)

Multimediální přehrávače - hudba, video, CD, DVD
Winamp, Windows Media Player, RealPlayer, WinDVD, PowerDVD *(pozn.: většina uvedených programů přehrává audio i video - zařazeny jsou podle nejčastějšího použití)*

Hudba (mp3, ogg, wma, CD)

- XMMS, Noatun (K), AlsaPlayer, Zinf, mpg123 (T), mpg321 (T), mp3blaster (T)

Video (MPEG I + II, DivX, DVD)

- MPlayer, VideoLAN Client, Xine, Ogle (pouze DVD), aKtion! (K), RealPlayer

Hudební nástroje: audio editory, MIDI sequencery, editory notace, ripování CD
CoolEdit, GoldWave, CDEX

Analog:

- Ecasound a Ecawave, Audacity, Kwave (K), GLAME (G), Ardour, ReZound, Aglaophone, Beast, DAP, Xwave
- GramoFile (T), wavnorm - nplay - nrecord (T), normalize (T), SoX (T)

MIDI:

- JAZZ++, Anthem, MidiMountain, Melys (G), MusE (K), MidiToy (K),

Notace:

- LilyPond (recenze), MusiXTeX, Notedit, Rosegarden (i sequencer), XGMC, KGuitar (K)

Ripování CD:

- RipperX (G), Grip (G)

Vypalovací programy
Nero, Easy CD Creator, CDRWin, FireBurner, DiscJuggler

- cdrtools (cdrecord, cdda2wav, mkisofs) (T), cdrdao (T)

Front-endy (nejen) pro cdrtools a cdrdao:

- arson (K), k3b (K), CD Bake Oven (K), Gnome Toaster (G), Coaster (G2), XCDRoast, ECLiPt Roaster (G)

HTML editory + univerzální textové editory
Notepad :), HotDog, Dreamweaver, GoLive

- Quanta (K), Screem (G), Mozilla Composer (CZ), Bluefish, CoffeeCup ($$$), ASHE
- NEdit, jEdit, Beaver (G), Vim (T), Emacs (T), XEmacs, joe (T)

IDE (Integrated Development Environment)
Delphi, Visual Studios

- Anjuta (G), KDevelop (K), QtDesigner, IDE Studio, IDEntify, jGRASP, QIDE, TIA (T), motor (T), RHIDE (T), Jabberwocky (Lisp), eric3 (Python)

Pascal:

- Lazarus, Kylix (Delphi pro Linux), FreePascal (T)

Java:

- Java DE for Emacs, netbeans

$$$:

- Code Crusader, Code Forge, CodeGuide, CodeWarrior

Instant Messaging
ICQ, Jabber, MSN, AIM, Yahoo, IRC, atd.

- Kopete (K) - ICQ, MSN, AIM, Jabber, IRC, Gadu-Gadu
- Everybuddy - ICQ, MSN, AIM, Jabber, IRC, Yahoo!
- Gaim - ICQ, MSN, AIM, Jabber, IRC, Yahoo!, Gadu-Gadu, Zephyr

- CenterICQ (T) - ICQ, MSN, AIM, Jabber, IRC, Yahoo!
  Pouze ICQ:
- GnomeICU (G), KXicq (K),SIM, Licq, mICQ (T), ickle
  Pouze MSN:
- harbinger, KMerlin, KMess, Msn4Lin
  Pouze Jabber (a tímpádem vše ostatní :)):
- Psi, Gabber, Iruka, Tkabber, JabberX (T), IMCom (T)
  Pouze IRC:
- KVIrc, KMyIRC (K), Xchat, LostIRC, BitchX (T)
  Pouze zbytek...
- Kadu - Gadu-Gadu, AIM for Linux, Yahoo! for Linux

Sdílení souborů, peer-to-peer sítě
Gnutella, Direct Connect, eDonkey, Kazaa

Gnutella:
- Gtk-Gnutella, LimeWire, Phex, Qtella, Mutella (T + web)

Direct Connect:
- Valknut, dc qt, QuickDC, DCTC (T), LDCC (T)

eDonkey:
- mldonkey, eDonkey for Linux (T)

Ostatní:
- PySoulSeek - SoulSeek
- giFTcurs - giFT

Chybí vám tu něco naprosto nepostradatelného? Poslední záchranou je víno: WINE a CrossOver. A kdyby bylo úplně nejhůř, pak: bochs, Win4Lin nebo Vmware.

Súborové systémy

Pravdepodobne viete, že údaje sa v počítači ukladajú na disk. Na to, abyste mohli ale nejaké údaje ukladať, musí byť jasné, kde presne na disku tieto údaje budú uložené a ako ich nájdete, až ich budete potrebovať. Presne túto úlohu riešia súborové systémy. Súborový systém je teda štruktúra obsahujúca súbory (ich obsah) a ich atribúty a ďalšie informácie potrebné pre prácu so súbormi - metadáta.

Čo majú spoločné adresár a súbor?

Adresár (angl. directory) je v našom jazyku známy pod mnohými menami: zložka, priečinok, atď. Treba si uvedomiť, že adresár je len zvláštny typ súboru. Jeho zvláštnosť spočíva v tom, že pre daný typ súborového systému má vopred definovanú štruktúru. Pre novšie, inteligentné súborové systémy táto štruktúra obsahuje meno súboru a číslo tzv. inodu. Inode je časťou metadát a obsahuje informácie o súbore ako sú jeho prístupové práva, dátum a čas vzniku, veľkosť a podobne. To prináša zo sebou zaujímavú možnosť mať v dvoch rôznych adresároch záznam ukazujúci na ten istý inode. Vtedy vlastne existuje jeden súbor na disku, má jediné miesto uchovávajúce prístupové práva, veľkosť a podobne, ale môže mať rôzne mená. Vo svete Unixu sa to označuje ako linka - hard linka.

Pripájanie a odpájanie súborových systémov

Na rozdiel od operačného systému MSDOS alebo Windows, súborové systémy v Unixoch spravidla nevidno automaticky (výnimku tvorí systém supermount, ktorý je ale mimo rozsah tohoto dokumentu). Každý súborový systém, ktorý má byť dostupný operačnému systému či jeho užívateľom, je nutné najprv pripojiť. Súbor `/etc/fstab` obsahuje záznam o tom, ktoré súborové systémy sa pripoja automaticky pri štarte systému.

Je dôležité si uvedomiť, že potreba pripájať a odpájať súborové systémy sa týka aj diskiet, CD-čiek a podobne. Zariadenia, ktoré podporujú zamykanie, zamknú médium po dobu, kedy je súborový systém na médiu pripojený. To znamená napríklad, že CD mechanika alebo ZIP mechanika nereagujú na tlačidlo pre vybratie média, ak súborový systém na tomto médiu je pripojený. Na druhej strane, súborový systém nie je možné odpojiť, ak sa používa. To znamená, že ak nejaký program má otvorený súbor na tomto súborovom systéme, alebo nejaký proces má adresár na tomto súborovom systéme nastavený ako aktuálny adresár.

Žurnálovacie súborové systémy

Ako sme už povedali, jednou zo základných vlastností súborového systému je uchovávanie údajov. A to najhoršie, čo sa môže stať, je havária, ktorá môže mať za následok stratu týchto údajov. Žurnálovacie súborové systémy boli vyvinuté s cieľom obmedziť riziko straty dát na minimum. Preto sa každá operácia zapisuje do takzvaného žurnálu a v prípade, že sa skutočne úspešne dokončí, tak sa v žurnále označí za vykonanú. V prípade, že dôjde k havárii, tak v žurnále môže systém zistiť, ktoré operácie boli dokončené, a čo bolo skutočne zapísané na disk. Operácie, ktoré neboli dokončené, akoby sa ani nestali. To znamená, že síce môžete prísť o nejaké údaje (z poslednej operácie), ale stav súborového systému ako takého je konzistentný. Za zmienku stojí informácia, že z dôvodou rýchlosti sa často žurnálujú len metadáta.

FAT

FAT (*F*ile *A*llocation *T*able) ste mohli stretnúť medzi prvými v operačnom systéme MS-DOS. Charakterizovať ho možno tým, že súbory usporadáva do stromovej štrukúry adresárov, pričom hlavný adresár má stanovený maximálny počet položiek, ktoré môže obsahovať. Ďalšou charakteristickou črtou je FAT tabuľka. FAT filesystém obsahuje tieto tabuľky dve a pokiaľ je všetko v poriadku, tak sú rovnaké.

FAT je blok dát na začiatku súborového systému, kde sú poradové čísla blokov, na ktorých sa nachádza ten ktorý súbor. V časoch MS-DOS 3.1 boli tieto čísla 12 bitové. Preto môžete občas nájsť tiež označenie FAT12. Samozrejme z toho vyplýva aj obmedzenie na počet dátových blokov, ktoré môže taký súborový systém obsahovať - 2^{12}.

S príchodom väčších diskov prišli aj FAT systémy so 16 bitovými indexami blokov (FAT16) a neskôr až 32 bitovými indexami (FAT32). Poslednou črtou, ktorú spomenieme, je obmedzenie mien súborov. Pôvodne bolo toto obmedzenie stanovené na 8 znakov vlastného mena a 3 znaky prípony. Oboje mohli obsahovať len ASCII znaky. Keď narástla potreba na dlhšie mená súborov, firma Microsoft pre udržanie spätne kompatibility zaviedla konvenciu, podľa ktorej je ôsmy znak nahradený znakom ~ (tilda), na znak toho, že meno súboru pokračuje v ďalšej položke adresára. Tieto rozšírenia dali príčinu pre nové meno: VFAT (Versatile(?) FAT). Nedostatkom FAT je, že informácie inodu má uložené priamo v štruktúre adresárov.

Jedným z miest, kde FAT prežíva do dnešných čias, sú diskety.

```
# mount /dev/fd0 /mnt/floppy/
```

Všimnite si ale, že prístupové práva v takomto prípade povoľujú prístup pre zápis len pre roota:

```
# ls -ld /mnt/floppy /mnt/floppy/subor.txt
drwxr--r--  3 root  root   7168 Jan  1  1970 /mnt/floppy
-rwxr--r--  1 root  root  92599 Jul 19  1996 /mnt/floppy/subor.txt
```

Pokiaľ chcete povoliť prístup iným užívateľom, môžete použiť niektorý z iných spôsobov. Všimnite si prístupové práva adresára

```
# mount -oumask=022 /dev/fd0 /mnt/floppy/
# ls -ld /mnt/floppy/ /mnt/floppy/subor.txt
drwxr-xr-x  3 root  root   7168 Jan  1  1970 /mnt/floppy/
-rwxr-xr-x  1 root  root  92599 Jul 19  1996 /mnt/floppy/subor.txt
# umount /mnt/floppy
# mount -oumask=007,uid=rastos,gid=floppy /dev/fd0 /mnt/floppy/
# ls -ld /mnt/floppy/ /mnt/floppy/subor.txt
drwxrwx---  3 rastos  floppy   7168 Jan  1  1970 /mnt/floppy/
-rwxrwx---  1 rastos  floppy  92599 Jul 19  1996 /mnt/floppy/subor.txt
```

Aby ste nemuseli vždy zadávať všetky parametre programu `mount`, môžete vložiť príslušný riadok do `/etc/fstab`:

```
/dev/fd0   /mnt/floppy     auto
 defaults,noauto,users,uid=rastos,gid=floppy,umask=007  0  0
```

NTFS a WinFS

NTFS je ďalším vývojovým krokom z dielne Microsoftu. Je rýchlejší, bezpečnejší a skutočne používa koncepciu inodov. Jeho hlavný nedostatok vidím v tom, že podrobnosti o jeho štruktúre nie sú verejne dostupné. Dôsledkom toho je to, že linuxový ovládač pre tento súborový systém podporuje len čítanie. Kód pre zápis je označovaný za experimentálny a v súčasnej dobe by mal umožňovať prepisovanie obsahu súborov za predpokladu, že nemeníte ich dĺžku.

WinFS vlastne nie je súborový systém, ale nadstavba nad NTFS, ktorá má umožniť rýchlejšie vyhľadávanie údajov pridaním istej funkcionality z oblasti databáz.

Pozor na to, že súbory, ktorých mená obsahujú znaky mimo ISO-8859-1, nemusí byť pod Linuxom vidno, ak nepoužijete príslušnú voľbu pri pripájaní (pre slovenčinu a jadrá 2.4 `iso8859-2`, pre jadrá 2.6 `nls=iso8859-2`)

```
# mount -oro,umask=0222,nls=iso8859-2 /dev/hda2 /mnt/nt
# ls -ld /mnt/nt "/mnt/hd/Documents and Settings/rastos/My Documents/"
dr-xr-xr-x  1 root  root   8192 2003-01-25 17:39 /mnt/nt
-r-xr-xr-x  2 root  root 630784 2003-01-25 18:46 /mnt/hd/Documents and
                                Settings/rastos/My Documents/návod.doc
```

Príslušný riadok v `/etc/fstab` potom vyzerá takto:

```
/dev/hda2 /mnt/nt ntfs defaults,ro,users,umask=0222,nsl=iso8859-2
1 0
```

Unixové súborové systémy

Svet Unixu a jeho klonov je ďaleko bohatší. Nájdeme tu

- UFS - Unix File System
- Minix File system
- ext2 - second extended filesystem
- ReiserFS
- a mnoho iných

Jednou z typických vlastností unixových súborových systémov je to, že blok inodov (tzv. superblok) má viacero kópií rozmiestnených po disku, ktoré operačný systém udržiava zosynchronizované. V prípade, že dôjde k pádu systému, môžete stanoviť, ktorá kópia sa má pužiť pri oprave súborového systému.

Keďže najpravdepodobnejšie sa stretnete s operačným systémom Linux, spomenieme situáciu na ňom. Ešte donedávna takmer každý linuxový systém používal `ext2`. Potom ale narástla ponuka žurnálovacích súborových systémov. Preto dnes môžeme bežne stretnúť `ext3` (nástupcu ext2) či `reiserfs`.

Súborový systém Ext2

Ext2 je jeden z najrozšírenejších súborových systémov používaných pod Linuxom. Je to klasický unixový súborový systém podporujúci uchovávanie prístupových práv vlastníka, skupiny a ostatných. Podporuje symbolické linky i hard linky, špeciálne súbory atď.

Vlastnosti súborového systému ext2 možno upravovať pomocou programu `tunefs`. Medzi vlastnosti patrí napríklad percento kapacity, ktoré je rezervované pre užívateľa `root`. Výchozí hodnota je 5 %. Ďalšou vlastnosťou je príznak, ktorý označuje, či je daný súborový systém *čistý* - teda či je konzistentný. Tento príznak sa nastaví na *nie* pri pripojení systému na zápis a na *áno* pri odpojení.

Ak dôjde k pádu operačného systému, tento príznak zostane nastavený na *nie* a podľa neho `fsck` vie, či má vykonať úplnú kontrolu alebo nie. (Prepínač `-f` programu fsck vynúti vykonanie kontroli aj keď súborový systém vyzerá čistý.) Poslednou vlastnosťou, ktorú spomeniem, je počet pripojení, po dosiahnutí ktorého bude súborový systém označený ako `not clean`, aj keď k žiadnemu problému neprišlo. Ak sa chcete dozvedieť o ďalších zaujímavostiach ext2, pozrite sa na program `dumpe2fs`.

Súborový systém UMSDOS

Tento súborový systém je zaujímavý tým, že je založený na FAT. Odlišuje sa tým, že dopĺňa do neho vlastnosti typických unixových súborových systémov, ako je udržiavanie informácií o vlastníkovi, skupine, prístupových právach a podobne. Tieto informácie sú uložené v súbore `--linux-.---`, ktorý existuje v každom adresári takéhoto súborového systému. Keď súborový systém UMSDOS pripojíte ako FAT, tak tam tento súbor vidíte. Keď ho pripojíte ako UMSDOS, tak systém tento súbor skryje, ale zapisuje do neho zmeny vo vlastníctve, prístupových právach a iné atribúty. V prípade, že sa niečo zmení v čase keď tento súborový systém nie je pripojený ako UMSDOS, možno údaje v súbore `--linux-.---` obnoviť programom `umssync`.

Tento súborový systém teda umožňuje použiť existujúci súborový systém FAT ako vlastný unixový súborový systém. To sa často používa pre vyskúšanie unixového systému bez potreby zriaďovať pre neho samostatný súborový systém. Jeho navýhodou však je nižšia rýchlosť.

Súborový systém ISO9660

Súborový systém ISO9660 je systém, ktorý sa typicky používa na dátových CDčkach. Hudobné CDčka majú inú štruktúru a nie je nutné (ani možné) ich pripájať ako súborový systém. Existujú tiež tzv. hybridné CDčka obsahujúce oblasť v formáte ISO9660, ako aj oblasť hudobnú.

So súborovým systémom sa teda stretneme najskôr keď potrebujeme pristupovať na CD. Základný ISO9660 má obmedzenia na mená súborov, hĺbku adresárovej štruktúry a podobne. Tieto obmedzenia obchádza napr. rozšírenie Joliet (MS). Ďalším rozšírením je El Torito, ktoré umožňuje bootovanie z CD-čka a ďalším je Rock Ridge, ktoré umožňuje ukladanie špeciálnych vlastností, ako sú symbolické linky a podobne. V Linuxe je potrebné mať toto rozšírenie zapnuté vo vlastnostiach ovládača pre ISO9660.

Pri napaľovaní CDčiek sa zvyčajne postupuje tak, že sa vytvorí obraz budúceho CDčka ako súbor pomocou programu `mkisofs` a ten sa potom napáli pomocou programu `cdrecord`. Samozrejme existuje niekoľko programov, ktoré ponúkajú peknú a šikovnú grafickú nadstavbu nad cdrecordom.

Loopback súborový systém

Ak máte v jadre podporu pre tzv. loopback súborový systém, môžete ako súborový systém pripojiť obraz iného súborového systému zapísaného do súboru. To možno použiť napríklad pre kontrolu obrazu ISO9660 predtým, než ho napálite na CD-čko.

```
# mkisofs -quiet -o test.iso dir/
# losetup /dev/loop0 /tmp/test.iso
# mount /dev/loop0 /mnt/tmp
```

Program `losetup` povie systému, že požiadavky systému na zariadenie `/dev/loop1` sa presmerúvajú na `/tmp/test.iso`. Toto presmerovanie by ste mali po skončení používania zrušiť.

```
# losetup /dev/loop0
/dev/loop0: [0303]:1006721 (/tmp/test.iso)
# umount /dev/loop0
# losetup -d /dev/loop0
```

Existuje tiež varianta loopback súborového systému, ktorá šifruje dáta, ktoré sa do neho zapisujú, a prečítať ich možno len po zadaní správneho hesla pri pripájaní - Cryptoloop. Vytvoriť takýto šifrovaný súborový systém môžete takto:

```
# dd if=/dev/random of=/file bs=1k count=100
# losetup -e losetup -e aes-256 /dev/loop0 /tmp/file
Password:
# mkfs /dev/loop0
# losetup -d /dev/loop0
```

Potom ho už možno používať:

```
# losetup -e aes-256 /dev/loop0 /tmp/file
Password:
# mount -oencryptionn=aes-256/dev/loop0 /mnt/tmp
Password:
...
# umount /dev/loop0
# losetup -d /dev/loop0
```

Sieťový súborový systém - NFS

Doteraz sme spomínali súborové systémy, ktoré sú fyzicky umiestnené priamo v našom počítači - disk, CD, súbor. Okrem toho ale existuje aj možnosť pripájania súborových systémov po sieti. Hovorí sa tomu NFS - **N**etwork **F**ile **S**ystem. Pri jeho použití treba vedieť, že existuje viacero verzií NFS, že reakcie na prácu so súbormi na NFS závisia na priepustnosti siete a tiež, že prenos dát nie je nijak zvlášť zabezpečený pred útočníkmi.

Pre úspešné používanie musíte mať na klientovi naštartovaný `rpc.portmap` a `rpc.mountd` a na serveri `rpc.nfsd`. Server musí tiež špecifikovať v súbore `/etc/exports`, ktoré adresáre ponúka.

```
# cat /etc/exports
/home clnt.domain.org
```

Tento server ponúka teda adresár `/home` stroju `clnt`.

```
# showmount -e srvr
Export list for srv:
/home clnt.domain.org
```

Teraz môžete pripojiť systém z druhého stroja:

```
# mount srvr:/home /home
```

Zdieľanie v sieti MS Windows - Samba

Ak v sieti máte počítače s MS Windows, pravdepodobne poznáte možnosť pripájania a zdieľania adresárov. Unixové systémy tiež podporujú túto funkcionalitu. Implementovaná je v balíku Samba. Pripájanie zdieľaných adresárov je možné v prípade, že máte v jadre podporu pre `smbfs` (a máte nainštalovanú sambu), pomocou programov `smbmount` (alebo špecifikovaním typu `smbfs` za `-t` po `mount`).

```
# smbmount -ousername=rastos \\\\srvr\\share /mnt/tmp
Password:
```

Swap

Moderné počítačové systémy majú pomerne veľké nároky na pamäť. Pretože diskový priestor je lacnejší než RAM pamäť a využitie všetkej RAM sa nestáva často, operačný systém môže odložiť časť obsahu pamäte, ktorá sa momentálne nepoužíva, na disk. MS Windows odkladá do súboru. Linux ponúka na výber súbor alebo vyhradenú partíciu. Tradične sa používa partícia, pretože je rýchlejšia.

Swap vlastne nie je súborový systém v pravom slova zmysle, pretože neumožňuje ukladanie súborov. Vytvoríme ho programom `mkswap`:

```
# mkswap /dev/hda5
Setting up swapspace version 1, size = 201240 kB
```

Používanie je riadené dvojicou programov `swapon` a `swapoff`:

```
# swapon /dev/hda5
...
# swapoff /dev/hda5
```

Aby sa swap použil automaticky pri naštartovaní systému, môžete mu vytvoriť položku v `/etc/fstab`:

```
/dev/hda5   none   swap   defaults   0   0
```

Poslednú otázku, ktorú treba pri používaní swapu vyriešiť, je rozhodnutie o jeho veľkosti. V starších dokumentoch sa dočítate odporúčanie, že by mal byť dvakrát taký veľký ako RAM pamäť. V skutočnosti je to trocha inak. Jediným správnym spôsobom je odhadnúť, koľko virtuálnej pamäte bude systém potrebovať pri svojej činnosti. Ak máte pracovnú stanicu, kde beží tabuľkový procesor či kompilácia menšieho projektu a má 512MB pamäte, je zbytočné nastavovať 1GB swap. Naopak, ak pobežíte veľkú databázu, nároky na pamäť môžu byť dosť veľké. Takže odporúčam vysledovať, koľko pamäte používajú aplikácie, ktoré bežne používate, a k tomu niečo pridať. Zvážte, že browser bežiaci mesiac v kuse môže potrebovať postupne viac, ako si zoberie krátko po naštartovaní. Zvážte, že za rok prejdete na novšiu veriu aplikácií či správcu okien a podobne a tomu prispôsobte svoj výpočet veľkosti swapu.

Tipy a triky pre súborové systémy pod Linuxom

Ako skontrolovať, či náš systém obsahuje podporu (driver) pre daný súborový systém v jadre?

```
# cat /proc/filesystems
nodev   sysfs
nodev   rootfs
nodev   bdev
nodev   proc
nodev   sockfs
nodev   usbfs
nodev   usbdevfs
nodev   futexfs
nodev   tmpfs
nodev   pipefs
nodev   eventpollfs
nodev   devpts
        ext3
nodev   ramfs
        msdos
        vfat
        iso9660
nodev   nfs
nodev   nfsd
nodev   cifs
        ntfs
        reiserfs
        udf
nodev   rpc_pipefs
```

Ak nie, budete musieť túto podporu pridať - buď nahraním príslušného modulu, alebo zakompilovaním príslušného driveru do jadra.

Pripájanie ako neprivilegovaný užívateľ

mount(8) - za normálnych okolností môže súborové systémy pripájať len užívateľ root. Ak súbor `/etc/fstab` obsahuje pre daný súborový systém voľbu `user`, môže ho pripojiť ktokoľvek, ale len ten, kto ho pripojil, ho môže odpojiť. Ak chcete, aby ktokoľvek mohol odpojiť nejaký súborový systém, dajte mu voľbu `users`.

Pripájanie len na čítanie

Voľba `ro`. Používaná napr. pre CD-čka.

Ako zistiť, čo sa dá pripojiť z iného servera cez NFS

```
showmount -e server
```

Zistenie typu súborového systému

Linux (a mnohe iné unixové systémy) ponúkaju program, ktorý sa volá `file`. Tento program používa informácie uložené v `/etc/magic`, aby uhádol, čo sa nachádza v nejakom súbore.

```
file - < /dev/hda6
standard input: Linux rev 1.0 ext3 filesystem data
```

Všimnite si použitie parametra – a presmerovanie. Bez nich by program povedal:

```
file /dev/hda6
/dev/hda6: block special (3/6)
```

Otvorenie a zatvorenie CD mechaniky programom

Program `eject` dá príkaz zariadeniu pre vyhodenie média. Ak na príkazovom riadku nezadáte inak, použije sa zariadenie `/dev/cdrom`. Ak je súborový systém na médiu pripojený, eject sa ho pokúsi najprv odpojiť. Ak mám pripojenú Iomega ZIP disketu (`/dev/sda`), môžem ju vysunúť príkazom

```
# eject /dev/sda
```

(ZIP disketa sa tvári ako disk. Má partície a zvyčajne obsahuje jeden súborový systém na `/dev/sda4`, ale príkazu `eject` musíte dat parameter zodpovedajúci celému zariadeniu, nie partície) Ak to dané zariadenie podporuje, tak prepínačom `-t` môžete zariadeniu povedať, aby si zobralo médium.

```
# eject -t /dev/cdrom
```

Ako zistiť, kam bežiaci systém swapuje

(robené na jadre 2.6.4)

```
cat /proc/swaps
Filename         Type     Size    Used    Priority
/dev/hda5     partition  196520     0       -2
```

Moderní souborové systémy 1.část

Úvod

Mnozí z Vás se jistě chtěli seznámit s pojmem souborový systém hlouběji. Ačkoliv si myslím, že se drtivá většina s pojmem souborový systém setkala, přesto ho nejprve objasním. Je to software, který slouží k organizování a používání dat uložených na záznamových médiích (pevný disk, CD apod). Souborový systém zajišťuje integritu dat (přeloženo do českého jazyka ucelenost). Tudíž informace, které uložíme, budeme moci později vyvolat v nezměněné podobě.

Pro svou činnost filesystém ukládá informace o souborech a informace o sobě samotném (vlastnictví, datum, kontrola přístupu, délka souboru a jeho lokace na disku apod). Bez těchto informací, tzv. *metadata*, by souborový systém nemohl pracovat. Nebudu se tu zabývat souborovými systémy jako ext2fs (běžně používaný souborový systém v OS Linux), o něm byla jistě již spousta materiálu sepsána a tak nebudu nosit dříví do lesa. Navíc nové filesystémy mají lepší vlastnosti a nemají tak přísná omezení jako třeba právě ext2fs.

Motivace nových souborových systémů

Modernější filesystémy použijeme zvlášť, pokud chceme zajistit integritu dat i při náhlém přerušení práce počítače. Existuje několik možností, co se při náhlém přerušení práce může stát:

- Systém stihl uložit soubor. Nic se neděje, můžeme pokračovat v práci.
- Systém nestihl uložit soubor (ani nezačal). Přišli jsme tedy o všechny změny, ale alespoň máme zachovánu starou verzi.
- Systém "spadl" během procesu ukládání. Toto je nejhorší případ! Dostáváme soubor, který se skládá částečně ze starého a částečně z nového. Když navíc zrovna zapisujeme metadata (jako třeba informace o adresářích), mužeme ztratit třeba celý adresář nebo dokonce data na diskovém oddílu (když se poruší metadata týkající se kořenového adresáře)!

Standardní linuxový souborový systém (ext2fs) svému poškození částečně předchází tím, že udržuje redundantní kopii metadat, takže se většinou nestává, že bychom o ně přišli. Pomocí kontroly integrity souborového systému (fsck), klasicky během bootování, je systém schopen zjistit, kde jsou metadata poškozená a nahradí je prostým zkopírováním redundantní verze. Nebo dojde ke smazání souboru, který byl přerušením poškozen. Samozřejmě že kontrola trvá tím déle, čím větší máme diskový oddíl a kontrola opravdu velkého disku může trvat velmi dlouho. Žádná z uvedených vlastností se samozřejmě nikomu z nás nelíbí, naštěstí existuje alternativa k těmto klasickým druhům filesystémů. Jsou jimi souborové systémy, které pracují s tzv. žurnálem.

Žurnálovací souborové systémy

O co se jedná. Stručně řečeno žurnálovací filesystém si uchovává informace o operacích, které provedl a je pak v případě výpadku schopen rychle se dostat zpět do konzistentního stavu. Změny jsou evidovány jako tzv. transakce. Jedná se o nezávislé atomické operace. Po každé transakci následuje potvrzení, když dojde k uskutečnění daného úkonu (např. zápis na disk). Proto pokud systém "spadne", můžeme najít v záznamech informace o provedených změnách a vrátit vše do původního stavu. Mezi tyto souborové systémy patří např. ext3, ReiserFS, XFS a JFS. V našem seriálu se postupně zmíníme o každém z nich podrobněji.

Omezení souborových systémů

Problémy při výpadcích však nejsou zdaleka jedinými nevýhodami tradičních souborových systémů jako ext2fs. Všechny byly navrženy v době, kdy záznamová média neměla takovou kapacitu jako v současné době. Dnes máme větší soubory, adresáře a také diskové oddíly a starší souborové systémy už nestačí ať už z hlediska různých omezení velikostí nebo výkonu. Tyto problémy jsou důsledkem interních struktur, na kterých jsou založeny. Hlavním problém je, že mají pevně danou délkou, což limituje jejich možnosti. Také metody, které v dřívější době vyhovovaly, jsou už při dnešních možnostech z hlediska výkonu nedostačující. Souborové systémy nové generace jsou navrhovány tak, aby problémům omezení předcházely.

V následující tabulce si můžete porovnat omezení jednotlivých souborových systémů:

Filesystém	Max. velikost filesystému	Velikost bloků	Max. velikost souboru
Ext2	4 TB	1KB-4KB	2 GB
Ext3	4 TB	1KB-4KB	2 GB
ReiserFS	16 TB	až 64KB	2^10 PB *1
XFS	18000 PB *1	512B - 64KB	9000 PB *1
JFS	512 B / 4 PB *2	512B, 1024B, 2048B, 4096B	512B / 512Tb *2

*1) 1PB = 10^15 B

*2) Maximální velikost souborového systému závisí na velikosti bloku dat, velikosti pro ostatní hodnoty dostaneme jednoduše pomocí trojčlenky.

Optimalizace výkonu nových souborových systémů

Vyhledávání volných bloků

Dalším problémem jsou struktury, pomocí kterých souborový systém hledá volné bloky při ukládání dat. Často se jedná o seznam, kde jsou udržována čísla jednotlivých volných bloků. UFS a ext2fs používá bitmapu, což je pole bitů, kde každý z nich odpovídá jednomu logickému bloku na diskovém oddílu. S narůstající kapacitou délka pole narůstá a výkon klesá.

Problémům, týkajících se hledání volných bloků dat, se vyvarujeme použitím tzv. "extents" a balancovaného stromu. Extents je skupina sousedících logických bloků, které jsou používány některými filesystémy. Deskriptor extents obsahuje 3 údaje:

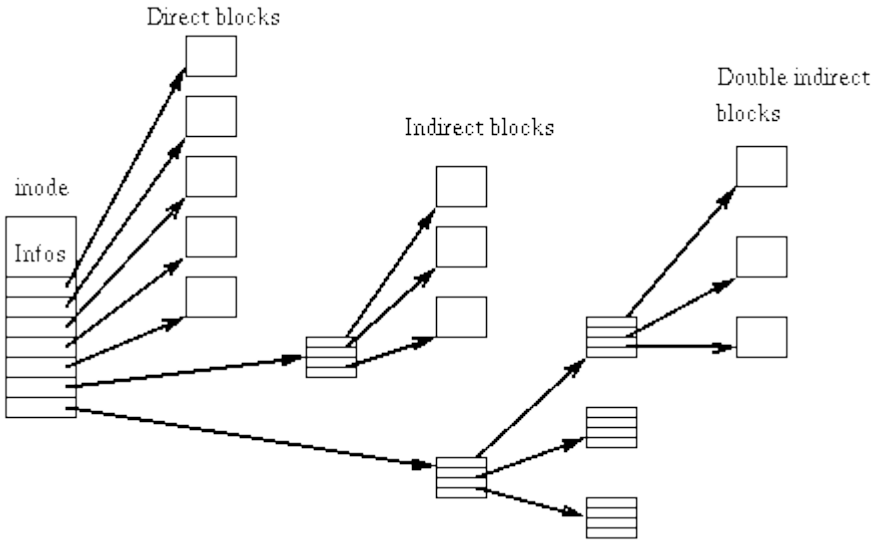
- adresa prvního bloku, kde extent začíná
- velikost v blocích
- údaj, který nám říká offset v rámci souboru, kde začínají data v daném záznamu uložená

V případě použití extents nezávisí velikost struktury, kde si uchováváme informace o volném místu, na velikosti filesystému. Také použitím balancovaného stromu místo prostého seznamu dochází k dalšímu zvýšení výkonu.

Problémy s velkými filesystémy

V případě velkého počtu položek adresáře je efektivita u starších souborových systému opět slabší. Často jsou položky adresáře ukládány do seznamu, a tudíž jejich následné vyhledávání je zbytečně zdlouhavé. Jedním z řešení je opět použití balancovaného stromu, kde jsou tyto položky uspořádány podle jmen.

Výkonnost však nedostačuje ani u velkých souborů. Pro vysvětlení této oblasti si musíme neprve objasnit pojem i-node. Jedná se o strukturu, kde souborový systém udržuje informace o souboru, jako jsou práva, typ souboru a hlavně ukazatele na bloky souborového systému, kde je soubor uložený. Obsahuje jednak přímé ukazatele, a také tzv. nepřímé ukazatelé, odkazující na bloky s ukazately přímými. Viz obrázek:



Problém spočívá v navržení struktury i-nody. Starší souborové systémy byly vytvářeny převážně pro práci s menšími soubory. Struktura i-nodů je proto ne zrovna efektivní. Čím větší soubor používáme, tím vícekrát přistupujeme k disku díky nepřímým pointrům. Důvodem, proč nepřímé pointry vůbec ext2fs používá, je to, že i-node má pevnou velikost.

Problém velkých souborů může být odstraněn použitím dynamického alokování i-nodů. Bohužel musíme vyřešit otázky jak zařídit mapování logických bloků i-nodu a jaké použít struktury na vyhledávání v rámci i-nodu. Řešením je většinou použití balancovaných stromů.

Řídké soubory

Omezení externí fragmentace a podpora řídkých souborů (sparse files) je také jednou z výhod nových filesystémů (ovšem jejich podpora je už v ext2fs). Pojem fragmentace většina čtenářů jistě zná, jen připomenu že se jedná o rozptýlení jednotlivých bloků souboru po disku a hlavička disku pak při čtení musí přejíždět z místa namísto. Samozřejmě výhodnější je mít bloky za sebou, operace pak budou rychlejší.

Řídké soubory jsou soubory, které vzniknou následujícím způsobem:

Zapíšeme například několik počátečních bytů dat a poté se chceme zapsat data uvnitř souboru, která odpovídají offsetu třeba 50000. Pokud nemá náš filesystém podporu pro řídké soubory, alokujeme všechny byty mezi počátkem a těmi bloky uvnitř souboru. Bloky mezi počátkem a offsetem 50000 nás však nezajímají a vůbec by alokovány být nemusely. Pokud použijeme souborové systémy s podporou řídkých souborů, pak se alokuje jen tolik místa, kolik zapisovaná data skutečně zabírají.

Moderní souborové systémy ext3

Úvod a motivace

V dnešním díle seriálu o souborových systémech v Linuxu se budeme zabývat filesystémem Ext3. Jedná se o filesystém s řadou výhod, ale také nedostatků. O nich bude právě řeč. Nejprve něco z jeho historie. Ext3 byl napsán Dr. Stephem Twediem jako nadstavba standardního linuxového souborového systému Linuxu (Ext2). Filesystém Ext2 je velmi rozšířený a oblíbený. Ačkoliv nemá podporu malých souborů jako ReiserFS nebo rychlost přístupu XFS, zůstává nejrozšířenější na linuxových platformách.

V dnešní době jsou některé Ext2 filesystémy opravdu velké, u nich pak trvá kontrola integrity dat v případě výpadku velmi dlouhou dobu (i několik hodin). Během doby, kdy se souborový systém kontroluje, jsou aplikace na počítači mimo provoz. Hlavním důvodem vzniku Ext3 je co nejvyšší dostupnost služeb i v případě výpadku. Nedisponuje ovšem vlastnostmi jako podpora extrémně rozsáhlých souborů, nebo mapování pomocí extents.

Jedním z nejdůležitějších cílů byla absolutní kompatibilita mezi Ext2 a Ext3 a to v obou směrech. Můžeme použít existující Ext2 filesystém, vytvořil žurnál a připojit ho jako Ext3 a dostáváme žurnálovací filesystém. Naopak pokud máme Ext3 filesystém a předtím došlo k bezpečnému odpojení tohoto souborového systému, můžeme ho připojit znovu jako Ext2 a vše funguje, jak má.

Jelikož Ext3 vychází z Ext2, má většinu jeho vlastností. Jako omezení velikosti souborového systému, velikosti souboru, velikosti bloků dat apod. Přestože některé informace se zapisují na Ext3 vícekrát než jednou, je Ext3 často rychlejší díky optimalizaci externí fragmentace.

Módy Ext3 filesystému

Pro optimalizování rychlosti a zajištění datové integrity můžeme zvolit jeden ze tří módů:

writeback
: zajišťuje jen limitovanou schopnost zachování integrity dat a za určitých okolností vede ke zvýšení rychlosti operací.

ordered
: je defaultním módem a garantuje konzistenci dat ve filesystému.

journal
: spotřebovává více prostoru na žurnál a proto opravení souborového systému po výpadku trvá delší dobu. Někdy je však rychlejší pro určité databázové operace, NFS a synchronní operace mail serveru

Writeback

Použití writeback módu dostáváme nejrychlejší variantu Ext3 filesystému. Je tomu tak, protože uchovává pouze změny metadat a nečeká dále na asociované změny dat souboru předtím než změní např. velikost souboru. Protože změny dat souboru a žurnálované změny metadat filesystému jsou uskutečněny nezávisle, mohou byt metadata v nekonzistním stavu (např. obsahují ukazatele na bloky dat, na které ještě nebyly data zapsány). Použitím tohoto módu získáme pouze urychlení kontroly konzistence dat při havárii systému.

Ordered

V módu ordered uchováváme opět pouze metadata souborového systému kvůli snížení redundance mezi zápisem do filesystému a do žurnálu a je proto rychlejší. Změny metadat jsou uskutečněny, narozdíl od módu writeback, až po změnách ve filesystému. Toto vede k lehkému snížení výkonu systému, naopak garantujeme, že při všech změnách budou metadata odpovídat datům, které jsme zapsali.

Journal

Použitím journal módu zapisujeme do žurnálu data i metadata. Každá operace se tedy musí uskutečnit dvakrát. Může proto snížit výkonnost systému, ale minimalizujeme šance ztráty nějakých dat.

Práce se souborovým systémem Ext3

Instalace

Nejprve potřebujeme zkompilovat kernel s podporou Ext3 filesystému. V jádrech řady 2.4 musíme zaškrtnout volbu CONFIG_EXT3_FS. Pokud používáte při konfiguraci `make menuconfig` nebo `make xconfig`, najdete tuto volbu v menu Filesystem, položka Ext3 journalling file system support. Podpora Ext3 filesystému se dá zkompilovat buď přímo do kernelu nebo jako modul. Pokud ovšem chcete používat Ext3 na kořenovém oddílu, doporučuji podporu Ext3 zakompilovat přímo do kernelu (viz dále).

Pokud máme existující Ext2 filesystém, můžeme ho lehce převést na Ext3 vytvořením žurnálu. Stačí použít následující příkaz:

```
tune2fs -j /dev/hdXX
```

, kde `/dev/hdXX` je cílový diskový oddíl. K převedení dokonce nemusíme ani daný filesystém odpojovat! Důležité je také přepsání záznamu v `/etc/fstab`, kde stačí prostě přepsat ext2 na ext3.

Pokud chceme vytvořit nový Ext3 filesystém použijeme:

`make2fs -j /dev/hdXX` , /dev/hdXX je cílové zařízení.

Nastavení

Jelikož Ext3 vychází z Ext2, sdílí spolu vlastnost, že e2fsck kontroluje pravidelně filesystém i v případě, že je označen jako nepoškozený. Oddíl se kontroluje buď při každém dvacátém připojení filesystému, nebo jednou za 180 dní (toto jsou defaultní hodnoty, které lze pomocí příkazu tune2fs změnit dle libosti). Jelikož je tato kontrola v případě Ext3 zbytečná a pravděpodobně chceme přejít na Ext3 právě z důvodu vyhnutí se fsck, dá se tato vlastnost vypnout použitím příkazu:

```
tune2fs -i 0 -c 0 /dev/hdXX
```

Pokud je náš kořenový souborový systém typu Ext3 a používáme kernel s podporou Ext3, systém ho připojí právě jako Ext3. Defaultnímu chování však můžeme zabránit následujícím parametrem bootloadru LILO:

```
LILO: linux rootfstype=ext2
```

stejně tak se dají nastavit parametry při připojování filesystému (jako žurnálovací mód):

```
LILO: linux rootflags=data=journal
```

Ext3 jako modul

Pokud máme podporu filesystému Ext3 zakompilovanou pouze jako modul, je připojení kořenového adresáře poněkud těžším problémem. Nejprve musíme vytvořit ramdisk, který bude obsahovat moduly zkompilované s kernelem. Je nutné zjistit přesnou verzi kernelu, který právě používáme. K tomu slouží příkaz:

```
uname -a
```

Dostaneme podobný vypis: Linux host **2.4.19** #2 Sat Oct ...

Pro samotné vytvoření ramdisku použijeme následující příkaz (vytváří ramdisk pouze s moduly potřebnými pro žurnálování):

```
mkinitrd --preload jbd --preload ext3 /boot/initrd-2.4.19.img 2.4.19
```

Příkaz `mkinitrd` získáme po nainstalování například balíčku initrd-tools v Debianu. Nyní stačí již jen upravit konfigurační soubor bootloaderu LILO, kde přidáme řádek pod image=... ,který říká systému, že má daný ramdisk načíst (viz manuálové stránky lilo.conf).

```
initrd=/boot/initrd-2.4.19.img
```

Poté můžeme systém restartovat a zkontrolovat výpis souboru `/proc/mounts`, jestli opravdu používáme ext3 filesystém. Jelikož toto je poměrně zdlouhavý proces, doporučuji zkompilovat kernel s přímou podporou Ext3.

Souhrn

Teď si ještě shrneme ostatní výhody souborového systému Ext3, o kterých jsem se zatím nezmínil. Mezi hlavní z nich patří podpora žurnálovacího souboru na odděleném diskovém oddílu. Zajímavá je také podpora ACL (Access Control Lists) a EA (Extended atributes), správa vadných sektorů a quot. Pomocí externího patche v kernelu dovoluje měnit velikost připojeného filesystému (ovšem pouze zvyšovat). V případě nepřipojeného souborového systému můžeme velikostí hýbat oběma směry. Pro ty, co by se o tuto problematiku dále zajímali, doporučuji: GNU Parted.

Moderní souborové systémy - ReiserFS

Úvod a motivace

V dnešním díle seriálu o souborových systémech se budeme bavit o historicky prvním linuxovém souborovém systému, který podporoval žurnálování. Jistě jste jméno ReiserFS už někdy slyšeli a v tomto díle se o něm dozvíte něco blíže. Hlavní osoba, která za tímto projektem stojí, je Hans Reiser, po kterém byl filesystém pojmenován. Ovšem vyvíjelo ho s ním samozřejmě ještě spousta dalších spolupracovníků.

Teď byste možná rádi věděli, proč ReiserFS používat. Má spoustu vlastností, které ostatní filesystémy neobsahují. Jak jsem již řekl, jedná se o žurnálovací souborový systém, tudíž jednou z výhod, které poskytuje, je zajištění konzistence dat při výpadku a také zrychlení obnovení systému oproti fsck.

ReiserFS je založen na rychlém balancovaném stromu. Je tak zrychlena práce např. s velkým množstvím položek v adresáři. Už není žádným problémem, aby adresář obsahoval třeba 100 000 souborů, ReiserFS s takto obsáhlým adresářem pracuje bez problémů.

Třetí a nejdůležitější vlastností je jeho schopnost pracovat efektivně s místem na diskovém oddílu. ReiserFS neukládá data do bloků pevných velikostí. Pokud chceme uložit několik souborů velkých například 100B, uloží jich ReiserFS filesystém několik do jednoho bloku. Použitím tohoto souborového systému tedy šetříme nezanedbatelnou měrou prostor na našem záznamovém médiu. Stejně tak jsou do jednoho bloku ukládány konce souborů, které už nezabírají celý blok. Na druhou stranu tím však dochází ke snížení výkonnosti filesystému, kvůli vyšší míře externí fragmentace.

Dalším problémem je přidání dalších dat na konec souboru. V tomto případě je na první pohled zřejmé, že se jedná o složitější problém, než např. u Ext2, kde přidaní několika bytů je triviální záležitostí. Jednoduše se použije nevyužité místo v rámci bloku, který používá soubor jen z části. Ovšem u Reiserfs musí dojít k přesouvání dat do jiných bloků, aby se tam přidávaná data vešla.

Uchovávání malých souborů se hodí hlavně v případě databází. ReiserFS byl navržen, aby umožnoval ukládat malé záznamy na disku, tudíž aby nebylo nutné používat nějaké způsoby shromažďování dat do větších souborů. Tento přístup by měl nejen zvýšit výkon, ale také snížit dobu vyvíjení aplikací tím, že programátoři nemusí vymýšlet způsoby, jak spojit několik malých bloků dat do jednoho velkého. Mohou je jednoduše na disk ukládat každý zvlášť.

Instalace

Jako první věc, kterou musíme provést, pokud chceme používat ReiserFS filesystem, je opět kompilace kernelu, který tento filesystém podporuje. Stačí povolit volbu `CONFIG_REISERFS_FS`. Pokud používáte `make menuconfig` nebo `make xconfig`, najdete ji v menu `Filesystems`, položka `Reiserfs support`. Podpora ReiserFS se dá zkompilovat přímo do kernelu nebo jako modul. Pokud budeme používat Reiserfs na kořenový svazek, je nutné zakompilovat podporu Reiserfs přímo do kernelu.

Pro vytvoření nového filesystému poté slouží příkaz:

```
mkreiserfs /dev/hdXX,
```

kde `/dev/hdXX` je zařízení, kde chcete nový filesystém vytvořit. Poté již můžeme bez problémů nový diskový oddíl připojit. Nevýhodou oproti ext3 je bohužel nemožnost přetransformovat filesystém z klasického ext2 na reiserfs bez ztráty dat.

Dále je nutné přidat záznam do konfiguračního souboru `/etc/fstab`, tento záznam může vypadat následovně:

```
/dev/hdc1 /home reiserfs defaults 0 0
```

Položka `/home` je místo, kam chcete diskový oddíl připojit, tudíž si ho samozřejmě můžete změnit podle potřeby.

Parametry zrychlující práci filesystému

Jak jsem již uvedl, ReiserFS má i některé své nedostatky (práce s konci souborů), které výkon snižují. Tyto nedostatky se však dají částečně napravit pomocí správných nastavení při připojování oddílu. Pomocí volby `noatime` zrušíme záznamy filesystému o posledním přístupu ke všem položkám filesystému. Další volbou, která zrychlí práci s filesystémy je `notail`. Tím zrušíme ukládání konců souborů do jednoho bloku. Ztratíme tím sice nějaké místo na disku, ale dojde k zrychlení práce filesystému. Tedy např. kořenový filesystém znovu přípojíme se se správnými parametry pomocí příkazu:

```
mount / -o remount, notail
```

Opravený záznam v souboru `/etc/fstab` je následující:

```
/dev/hdc1 /home reiserfs noatime,notail 0 0
```

Změna velikosti filesystému

Užitečnou vlastností ReiserFS filesystému je možnost změny jeho velikosti. Příkaz `resize_reiserfs` slouží právě ke změně velikosti diskového oddílu. Ke zvýšení velikosti ho dokonce nemusíme ani odpojovat. Pokud souborový systém zvětšujeme, je nutné, aby bylo samozřejmě na disku volné místo (např. zrušením následujícího oddílu pomocí příkazu `cfdisk`). Když chceme zmenšit filesystém, musíme nejprve použít `resize_reiserfs` a poté teprve `cfdisk` (pozor aby jste ho nesnižovali na menší než velikost, kterou data na něm zabírají!).

Například kdybychom chtěli zmenšit velikost filesystému na diskovém oddílu `/dev/hda2` připojeném na adresář `/mnt`, postupovali bychom následovně:

```
umount /mnt
resize_reiserfs -s -1G /dev/hda2
mount /dev/hda2 /mnt
```

Bližší informace získáte z manuálové stránky `resize_reiserfs(8)`.

Shrnutí

K dalším výhodám souborového systému ReiserFS patří podpora žurnálovacího souboru na odděleném diskovém oddílu a dynamické alokování inodů. Jak jsem se již zmínil také možnost online zvětšit filesystém. Zajímavá je podpora špatných bloků a pomocí externího patche i možnost quot.

Moderní souborové systémy - XFS

Úvod

V dnešním díle seriálu o moderních souborových systémech se budeme zabývat XFS filesystémem. XFS byl vyvinut firmou SGI jako náhrada filesystému EFS a v roce 2001 vyšla podpora tohoto filesystému i pro operační systém Linux. Tento souborový systém se používá pro high-end servery (obsahuje podporu pro multiprocesorové počítače), zajišťuje rychlé zotavení při pádech a podporuje extrémně velké diskové farmy. Jedná se o žurnálovací filesystém s 64-bitovým adresováním. Jeho hlavními výhodami jsou robustnost, důvěryhodnost a možnosti, které ostatní filesystémy

neposkytují. Má však i svoje nevýhody, jako je slabší výkon při mazání velkého počtu malých souborů nebo příliš veliký kód.

Mezi výhody XFS patří jeho následující vlastnosti:

- **žurnálování**, tudíž jeho rychlá obnova při havárii systému (dochází k žurnálování pouze metadat).
- **rychlé transakce**, kde se XFS snaží co nejvíce snížit vliv žurnálování při zápisu a čtení dat. Jeho struktury a algoritmy jsou navrženy tak, aby se tento cíl dařilo co nejvíce splňovat.
- **vysoké limity filesystému** vycházejí z faktu, že se jedná o 64-bitový filesystém, tudíž je schopen obsahovat soubory, které mají miliony terabytů. Tento filesystém je tedy připraven na vzrůst kapacity disků a to nejen svými vysokými limity, ale také strukturami a algoritmy.
- **vysoký výkon**, díky schopnosti efektivně používat I/O operace.

Instalace

Kompilace kernelu

Jako první věc, kterou musíme provést, pokud chceme používat XFS, je opět kompilace kernelu, který tento filesystém podporuje. Bohužel je v případě XFS tato procedura o něco zdlouhavější, protože podpora XFS není přímo ve zdrojových kódech kernelu připravena a musíme proto kernel tzv. opatchovat. Vzhůru do toho!

Začneme stažením zdrojových kódů kernelu (linux-2.4.x.tar.gz nebo linux-2.4.x.tar.bz2) z ftp.kernel.org nebo z nějakého mirroru a rozbalíme je do adresáře, kde budeme náš kernel kompilovat. Poté je nutné stahnout patche (je nutné stáhnout patch pro správnou verzi kernelu!), pomocí kterých upravíme zdrojové kódy kernelu tak, aby obsahovaly podporu XFS. Nyní stačí rozbalit patche pomocí příkazu `gunzip patchfile_name.gz` a zkopírovat patch do adresáře se zdrojovými kódy kernelu. Poté v tomto adresáři použijeme příkaz:

```
patch -p1 < patchfile_name
```

kterým opatchujeme zdrojové kódy kernelu. Nyní, když už jsme aplikovali patch, kofigurujeme a instalujeme kernel jako obvykle. Pro podporu XFS je nutné zaškrtnout tyto volby:

- `XFS filesystem support (CONFIG_XFS_FS)`
- `Page Buffer support (CONFIG_PAGE_BUF)`

Tyto volby můžete zaškrtnout buď jako modul, nebo přímo zakompilovat do kernelu. Pokud plánujete použít XFS na váš kořenový oddíl, je nutné mít podporu zakompilovanou přímo do kernelu (nebo můžete vytvořit ramdisk s tímto modulem viz druhý díl tohoto seriálu).

Nástroje na obsluhu XFS

Pro práci s XFS filesystémem jsou také nutné určité uživatelské nástroje, o kterých si něco řekneme právě teď. Tyto nástroje získáme na následující adrese: http://www.xfs.org/download.html.

Příkazy:

- **`acl`** - příkaz na správu ACL (Access control lists).
- **`attr`** - příkaz pro manipulaci s EA (extended attributes).
- **`quota`** - balíček příkazů pro monitorování a omezování používaného místa na disku uživatelem (skupinou uživatelů).

- **xfsdump** - tento balíček obsahuje příkazy `xfsdump`, `xfsrestore` a další příkazy pro administraci XFS. Příkaz `xfsdump` prohlíží soubory filesystému, poté rozhodne, které je nutné zálohovat a zkopíruje je na určený disk, pásku nebo jiné zařízení. Umí ukládat i EA (Extended attributes) a to ve formátu, který je vhodný na přenášení dat mezi různými architekturami. Příkaz `xfsrestore` provádí přesně opačnou činnost, obnovuje data za zálohy.
- **xfsprogs** - balíček příkazů pro prací s XFS, který obsahuje i `mkfs.xfs` (pozor: pokud máte starší verzi těchto příkazů, musíte si je překompilovat, aby seděly s novými hlavičkovými soubory kernelu!). Po rozbalení použijeme známou sekvenci příkazů `make configure; make; su root; make install.`

Vytvoření XFS filesystému:

Nový souborový systém XFS se vytváří stejným způsobem jako ostatní a to příkazem

```
mkfs -t xfs /dev/hdXX
```

kde `/dev/hdXX` je diskový oddíl, na kterém chceme XFS vytvořit. Při vytváření filesystému dochází k smazání všech dat, které se na daném oddíle vyskytují, proto je nutné si je zálohovat!

Existuje možnost zlepšení výkonu pomocí zvýšení velikosti žurnálu. Následujícím příkazem vytvoříme filesystém s defaultní hodnotou žurnálu 8000 bloků.

```
mkfs -t xfs -l internal,size=8000b -d name=/dev/hdXX
```

O dalších volbách, které je možné použít při tvorbě filesystému, se můžete dočíst v manuálových stránkách.

Nyní už stačí pouze připojit náš filesystém příkazem

```
mount -t xfs /dev/hdXX /adr
```

kde `adr` je místo v adresářovém stromu, kam chceme diskový oddíl připojit. Před připojením XFS zkontroluje transakce v žurnálu a pak můžeme náš nový filesystém začít používat.

Abychom nemuseli po každém restartu systému tento oddíl znovu připojovat ručně, je lepší přidat záznam do souboru `/etc/fstab` a systém při startu připojení provede za nás. Přidaná řádka by měla vypadat asi následovně:

```
/dev/hdc1 /home xfs defaults 0 0
```

Souhrn

XFS je moderní souborový filesystém s garantovanou konzistencí dat (díky žurnálu). Jeho podpora existujeje pro jádra řady 2.4 a 2.5 pomocí externího patche. Obsahuje podporu quot, EA (extended attributes) a ACL (access control lists). Stejně jako u všech ostatních moderních souborových systémů je zde možnost ukládat žurnál na oddělený diskový oddíl. Je možné za běhu zvyšovat velikost pomocí příkazu **xfs_growfs** z balíčku příkazů xfsprogs. Je kompatibilní s NFS. Podporuje také zálohování (viz příkazy **xfsdump** a **xfsrestore**). Dále je podporováno swapování do souborů. Pro běh tohoto filesystému je doporučováno minimálně 64MB paměti.

Moderní souborové systémy - JFS

Úvod

Po delší pauze způsobené nemocí a následném nedostatku času v průběhu zkouškového období je tu opět další díl seriálu o moderních souborových systémech. Dnes se podíváme blíže na poslední filesystém, který tento seriál hodlá pokrýt - JFS, a poté jednotlivé souborové systémy porovnáme. JFS je 64bitový filesystém vyvinutý světoznámou firmou IBM s důrazem na vysokou propustnost a spolehlivost hlavně na serverech. Své místo má hlavně na strojích s transakčně orientovanými operacemi.

Mezi jeho vyhody patří:

- **žurnálování**, tudíž jeho rychlá obnova při havárii systému
- **použití extendů**, díky čemuž dochází ke zrychleni práce filesystému, který produkuje efektivní a malé struktury pro mapování souborů.
- **různé velikosti bloků**, jelikož JFS podporuje bloky o velikostech 512, 1024, 2048, 4096 bytů, čímž dovoluje uživatelům zoptimalizovat výkon systému.
- **dynamické alokování inod**, čímž se zamezíme rezervování fixního místa na disku pro inody v průběhu vytvoření filesystému
- **dva způsoby organizace adresářů**. První z nich slouží pro malé adresáře (do 8 položek), obsah adresáře je uložen v inodě příslušného adresáře. Druhou variantou je B+ strom setříděný podle jména, který poskytuje daleko rychlejší možnost přístupu v porovnání se nesetříděnou organizací.
- **podpora řídkých souborů.**
- **podporuje velké soubory a souborové systémy**, což vyplývá z faktu, že se jedná o 64bitový filesystém, tudíž všechny položky struktur filesystému mají velikost 64bitů.

Instalace

Kompilace kernelu

Pokud nepoužíváte kernel, který podporuje filesystém JFS, je nutné si takový pořídit. JFS patche jsou k dispozici na adrese: http://oss.software.ib.com/jfs. Ke zprovoznění JFS budete potřebovat jak patche, tak programy na správu tohoto souborového systému (jfsutils), které najdete na stejné adrese.

Po stažení a rozbalení souborů se zdrojovými kódy kernelu získáme z výše uvedené adresy patche, např. `jfs-2.4-1.1.1-patch.tar.gz`. Tento soubor rozbalíme (doporučuji si přečíst README) a aplikujeme příslušné patche následujícími příkazy (z adresáře, do kterého jsme rozbalili zdrojové soubory kernelu):

```
patch -p1 < /usr/src/jfs/jfs-2.4-common-1.1.1.patch
patch -p1 < /usr/src/jfs/jfs-2.4.17-1.1.1-patch
```

Nyní, když už jsme aplikovali patch, nakonfigurujeme a instalujeme kernel jako obvykle. Pro podporu JFS je nutné zaškrtnout příslušnou položku v sekci Filesystems. Tyto volby můžete zaškrtnout buď jako modul, nebo přímo zakompilovat do kernelu. Pokud plánujete použít JFS na váš kořenový oddíl, je nutné mít podporu zakompilovanou přímo v kernelu (nebo vytvořit ramdisk s tímto modulem viz druhý díl tohoto seriálu)

Nástroje na obsluhu JFS

Pro práci s JFS filesystémem jsou také nezbytné nástroje na jeho správu zmíněné již výše. Instalaci těchto nástrojů zvládne jistě každý a to sekvencí příkazů:

```
./configure; make; make install
```

Patří sem příkazy na tvorbu (`jfs_mkfs`) a kontrolu (`jfs_fscklog`) filesystému, nastavení (`jfs_tune`) a další jako `jfs_fscklog`, `jfs_logdump` a `jfs_debugfs`.

Vytvoření JFS filesystému:

Nový souborový systém JFS se vytváří následujícím příkazem: `mkfs -t jfs /dev/hdXX`

kde `/dev/hdXX` je diskový oddíl, na kterém chceme JFS vytvořit. Při vytváření filesystému dochází k smazání všech dat, které se na daném oddíle vyskytují, proto je nutné si je zálohovat!

Nyní už stačí pouze připojit náš filesystém příkazem

```
mount -t jfs /dev/hdXX /adr
```

kde `adr` je místo v adresářovém stromu, kam chceme diskový oddíl připojit. Před připojením JFS zkontroluje transakce v žurnálu a můžeme náš nový filesystém začít používat.

Abychom nemuseli po každém restartu systému tento oddíl znovu připojovat, je lepší přidat záznam do souboru `/etc/fstab`. Všechny tyto operace jsou naprosto obdobné jako v případě všech ostatních souborových systémů.

Souhrn

JFS je tedy moderní žurnalovací souborový filesystém s garantovanou konzistencí dat (díky žurnálu). Je šířen pod licencí GPL, snaží se být portovatelný bez zásahu do zdrojových kódů kernelu a to pod všemi platformami, které podporuje Linux. Jeho podpora je zajištěna pro jádra řady 2.4 a 2.5 pomocí externího patche.

Srovnání souborových systémů

Kritéria výběru

Filesystémy můžeme porovnávat podle různých kritérií. Jak jste si mohli všimnout během našeho seriálu, nezáleží vždy na 100 procent na rychlosti. Mezi kritéria výběru toho správného filesystému patří následující vlastnosti:

- **stabilita** - provoz beze ztráty dat.
- **požadavky provozu** - samozřejmě nás nejvíce zajímají minimální požadavky na hardware pro běh filesystému.
- **kapacita** - limity filesystému (souborů, adresářů, vlastního souborového systému)
- **snadnost obsluhy** - zde nás zajímá skutečnost, jestli je během údržby filesystému nutné přerušit běh systému nebo poskytované služby.
- **možnost nastavení** - poskytuje filesystém možnost změny parametrů, které mají vliv na běh systému.
- **podpora** - dochází k dalšímu vývoji?

Tyto všechny vlastnosti určují vhodnost či nevhodnost souborového systému, nicméně výběr závisí hlavně na specifických vlastnostech použité platformy a také na druhu softwaru, který hodláme používat. Z toho vyplývá, že není možné nalézt optimální benchmark pro všechny druhy nasazení. Nejvěrnějších výsledků dosáhneme vlastními testy na svém hardwaru a používáním své aplikace.

Používané benchmarky

Schopnosti filesystému na vlastním stroji můžeme otestovat nejen pozorováním rychlosti běhu používaných aplikací, ale také některým z následujících benchmarků:

- **postmark** - soustředí se na otevírání, čtení/zápis, mazání a hledání.
- **iostone** - velké množství souborů, otevírání, čtení/zápis a zavírání souborů.
- **iozone** - detailní zhodnocení náhodného a sekvenčního přístupu k datům (čtení/zápis).
- **tiobench** - více vláken, propustnost a zpoždění

Tabulka vlastností

Feature	Ext3fs	ReiserFS	XFS	JFS
Firma/autor	Stephen Tweedie	Hans Reiser	SGI	IBM
Domovska stranka	link	link	link	link
Podpora v jadre od	2.4.15	2.4.1	2.5.36	2.5.6
Zurnal na oddelenem diskovem oddilu	ano	ano	ano	ano
Korenovy diskovy oddil	ano	ano	ano	ano
Dynamicky alokovane inody	pomoci externiho patche	ano	ano	ano
Zmena velikosti filesystemu za behu	jen zvetseni	jen zvetseni	jen zvetseni	jen zvetseni
Quoty	ano	ne	ano	externi patch
Zmena velikosti filesystemu	zvyseni i snizeni	zvyseni i snizeni	zvyseni	zvyseni
Extended atributes a Access control lists	externi patch	ano	ano	ne
Podpora rozptylenych souboru	ne	ano	ano	ano
Vyhledavani volnych bloku	sekvencni vyhledavani	B+ Tree	B+ Tree	binarni strom + bitmapa
Maximální počet souborů	4GB	4GB	4GB	4GB
Maximální počet souborů v adresáři	4GB	2GB	4GB	2GB
Maximální počet podadresářů	32000	64,5K	4G	65533
Maximální velikost souboru	4TB	16TB	16TB	16TB
Maximální velikost filesystému	16TB	16TB	16TB (až 9EB)	16TB (až 16PB)

pozn. všechny uvedené limity jsou pro 32-bitové architektury s velikostí bloku rovnou 4KB.

Závěr

Všechny dnešní moderní filesystémy se soustřeďují na co nejvyšší spolehlivost, snaží se udržet integritu dat za každé situace. A všechny z nich podporují následující vlastnosti:

- žurnálování/logování diskových operací
- rychlé zotavení filesystému po havárii
- podpora velkých souborů, diskových oddílů atd.
- flexibilní a sofistikované struktury metadat

Naopak jednotlivé filesystémy nabízejí některé své jedinečné vlastnosti jako například efektivní práce s malými soubory u ReiserFS nebo jednoduchost a kompatibilita Ext3 vzhledem ke standartnímu souborovému systému, který v Linuxu používáme - Ext2. Neexistuje proto nějaký "nejlepší" filesystém, ale pouze nejlepší pro danou úlohu. S jeho nalezením Vám již bohužel neporadím, nicméně doufám, že jsem Vám tímto seriálem při hledání toho pravého alespoň trochu pomohl.

Nebojíme se kompilace - I (teorie)

Tento seriál je určen začátečníkům, kteří neví, co je to překlad, případně neví jak na něj. Některé popisy jsou proto značně zjednodušeny.

V GNU/Linuxu, stejně jako v dalších unixových systémech, se velmi často používají překladače z rodiny gcc, kterými se také budeme v našem seriálku zabývat.

Překladač provádí přeměnu zdrojového kódu čitelného člověkem na binární kód, který je čitelný pro procesor. Ovšem čitelný pro konkrétní procesor. To, co umí i386, neumí Alpha a podobně.

Nutnost kompilace vznikla postupným vývojem. V dřevních dobách programování psali programátoři svá díla přimo ve strojovém kódu. Neměli jinou možnost, pokud chtěli, aby procesor věděl, co má dělat. Strojový kód je však pro větší projekty nevyhovující kvůli tomu, že zvládá pouze atomární kroky a složitější funkce je třeba velmi rozvlekle rozepisovat. Proto vznikly takzvané vyšší jazyky, které mají proti strojovému kódu řadu výhod:

- Jsou lépe čitelné pro člověka.
- Jsou na první pohled lépe srozumitelné.
- Díky tomu se v nich lépe hledají chyby.
- Programátor se může více soustředit na obsah než na formu.
- Překladač je pak může přeložit pro více platforem a systémů.
- Lze je dodatečně optimalizovat pro konkrétní procesor.

Toto jsou některé důvody, které programátory vedou k použití vyšších jazyků. V těch napíší své programy a pustí je do světa. Protože se pohybujeme ve světě svobodného software, máme ke všem aplikacím zdrojový kód. Ten nám umožňuje programy zkoumat, opravovat a měnit. Abychom je poté mohli na svých počítačích spustit, musíme provést zmíněnou kompilaci.

Samostatnou kapitolou je pak optimalizace programu. Ta nám umožní během překladu zohlednit charakteristiky jednotlivých procesorů, a tím zrychlit následné provádění strojového kódu. Příklad: procesory novějších generací obsahují instrukce, které umožňují provádět rychleji některé matematické funkce. Pokud je tedy v binárním programu kompilátor použije, zvýší se jeho efektivita. Nevýhodou pak je, že takto přeložený program nám nepoběží na procesoru nižší generace.

Programátora ovšem výsledná podoba nezajímá, tu určuje až překlad (respektive parametry předané překladači).

Tvůrci distribucí tyto překlady obvykle provedou za nás a my už získáme hotový binární (zkompilovaný) balíček, který jen rozbalíme na svůj disk pomocí instalačního procesu a můžeme začít program používat. Může však nastat několik situaci, které nám tento běžný postup znemožní:

- Potřebujeme program nějak upravit a musíme proto zasáhnout do zdrojových kódů.
- Neexistuje balíček pro naši distribuci.
- Chceme provést vyšší stupeň optimalizace.
- Potřebujeme balíček pro nestandardní architekturu.
- Balíčky, které máme, nejsou s knihovnami v naší distribuci kompatibilní.

Poslední možnost si popíšeme podrobněji, protože se nás pravděpodobně budou týkat nejčastěji. V moderním operačním systému provádí většina programů velmi podobné operace. Aby se každý programátor nemusel zabývat implementací obecných funkcí, existují takzvané knihovny. Tyto knihovny obsahují nejčastěji používané kódy, které může libovolný program použít. Jednoduše zavolá funkci v knihovně a ta nějak samostatně proběhne. Programátora tedy implementace vůbec nezajímá, stačí mu, že ví jak funkci zavolat.

Knihovny se do systému instalují samostatně (obvykle bývají v samostatných balíčcích) a bez nich pochopitelně programy fungovat nemohou. Bez nich by prostě kus kódu chybělo a program by nemohl některé funkce vůbec provádět. Správné knihovny ve správných verzích jsou tedy bezpodmínečnou nutností. Tady narážíme na zakopaného psa.

Nevýhoda knihoven se projeví ve chvíli, kdy se například snažíte do starší distribuce nainstalovat novější balíček. Tento balíček byl samozřejmě přeložen v novější verzi distribuce, která obsahuje také novější verze knihoven než ta vaše. Říkáme, že je program přeložen proti novější knihovně. Jelikož se jedná o binární balíček, je již pevně sestaven a očekává, že bude mít k dispozici knihovnu, proti které byl přeložen.

A nastává klasický problém závislostí, o kterém jste již jistě slyšeli. I kdyby se vám povedlo balíček nainstalovat, program nepoběží, protože nebude schopen s novou knihovnou komunikovat.

Chceme-li zjistit, které knihovny program potřebuje, můžeme použít příkaz `ldd`:

```
#ldd /bin/sh

libtermcap.so.2 => /lib/libtermcap.so.2 (0xb75df000)
libdl.so.2 => /lib/libdl.so.2 (0xb75db000)
libc.so.6 => /lib/tls/libc.so.6 (0xb74a3000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0xb75eb000)
```

V tomto případě je vše v pořádku a my vidíme, že žádná knihovna nechybí (v opačném případě bychom dostali místo cesty jen text "not found").

Jak z toho ven? Možností je několik:

- Nainstalovat novou knihovnu.

To ovšem nemusí být řešení - nová knihovna může být závislá na dalších knihovnách a ty pak na dalších a dalších a problém může přerůst až ve změnu půlky systému.

- Sehnat si balíček nového programu přeložený proti mojí distribuci.

Toto je samozřejmě ideální řešení, protože překlad už někdo udělal přede mnou. Ne vždy je to však možné.

- Zkompilovat si nový program vlastními silami.

Kompilaci vlastními silami si podrobně vysvětlíme a předvedeme na příkladu. Není to nic těžkého, protože člověk je tvor líný a vynašel si k tomuto účelu řadu automatických udělátek.

Ale o tom až příště.

Nebojíme se kompilace - II (praxe)

V minulém díle našeho miniseriáliu jsme si představili kompilaci teoreticky. Řekli jsme si, o co se vlastně jedná, k čemu je to dobré a dnes si řekneme, jak na to.

Nástroje

Už jsem vysvětlil, že se jedná o plně automatickou činnost řízenou jistými utilitami, které jistě najdete i ve své distribuci. Jde především o:

gcc

gcc je překladač, o kterém jsme mluvili v minulém díle. Provádí samotnou kompilaci, optimalizaci a všechny ostatní důležité operace.

make

Make je velmi důležitou utilitou, která řídí celou kompilaci. Vznikla z potřeby jednoduše zpracovat i velmi rozsáhlé projekty, jejichž ruční kompilace by byla velmi náročná. U větších projektů je potřeba zkompilovat velké množství dílčích souborů, provádět různé úpravy a podobně. Make toto vše dělá automaticky podle souboru nazvaného `makefile`.

Tento soubor se musí nacházet v adresáři s kompilovaným projektem. Obvykle se vytváří až na místě dle konkrétního systému.

Knihovny

Další důležitou součástí nutnou pro kompilaci jsou takzvané vývojářské knihovny. Ty obsahují hlavičkové soubory nutné pro linkování s konkrétní knihovnou. Od běžné knihovny se obvykle liší přívlastkem `dev` nebo `devel`, který je součástí názvu. Knihovna a vývojařská knihovna se tedy mohou jmenovat například:

```
xlibs-4.1.0
xlibs-dev-4.1.0
```

Pro běh programu je potřeba mít nainstalovanou jen uživatelskou část, kdežto pro překlad potřebujeme i vývojovou část. Ta ale obvykle není příliš velká, takže se o místo na disku bát nemusíte a není třeba ji po kompilaci odinstalovávat.

Pokud na některou vývojařskou knihovnu zapomenete, překlad neproběhne a poměrně srozumitelně se dozvíme, co je potřeba udělat, abychom chybu napravili.

Zdrojové kódy

Nyní se už dostáváme k samotnému překladu. Před samotným začátkem si neodpustím malou připomínku: **Překlad jako takový nemusíte provádět jako root a také to nedoporučuji!**

Prvním krokem je samozřejmě stažení souboru obsahujícího samotné kódy překládaného programu. Ten obvykle najdete na domovské stránce konkrétního projektu. Nemůžete-li program najít, obraťte se na Google případně rovnou na Freshmeat.

Stažený soubor (obvykle sbalený tarem a komprimovaný gzipem) rozbalíme pomocí příkazu

```
tar -xzf nas_zdrojak.tgz
```

Dalším a často opomíjeným krokem je přečtení dokumentace. Obvykle se v nejvyšším adresáři s kódy nachází soubor INSTALL nebo README. Přečtěte si jej! Obsahuje postup kompilace, která se může od námi probíraného mírně lišit. Obvykle také obsahuje seznam knihoven potřebných ke kompilaci a běhu programu.

configure

Většina kompilací probíhá velmi podobně. Nejprve je potřeba vygenerovat pracovní soubor pro make. Jeho generování probíhá až na cílovém stroji právě proto, aby bylo možno překlad uskutečnit na různých distribucích, platformách a dokonce systémech.

K samotnému vygenerování slouží obvykle skript `configure`. Ten se nachází v kořenovém adresáři kódů a často pomocí něj můžeme ovlivnit samotnou kompilaci. Volitelně totiž přijímá kompilační parametry, které mění výslednou podobu makefile a tím i konečného programu. Napíšeme tedy

```
cd nas_zdrojak
./configure --help
```

a dozvíme se, co a jak, případně jaké parametry můžeme použít. Dejme tomu, že chceme, aby program uměl běhat pod X serverem (a tedy v grafice) a vybrali jsme si parametr `--enable-gui`. Můžeme tedy zadat

```
./configure --enable-gui
```

a skript se spustí. Obvykle je psán v co nejuniverzálnějším jazyce a postupně si ohmatá celý váš systém a pokud najde vše potřebné, vygeneruje zmíněný soubor makefile.

Během práce uvidíte výpis podobný tomuto (zkráceno):

```
checking for gcc... gcc
checking for C compiler default output... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ANSI C... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking for g++... g++
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
```

```
checking dependency style of g++... gcc3
checking for a BSD-compatible install... /usr/bin/install -c
checking for ranlib... ranlib
checking how to run the C++ preprocessor... g++ -E
checking for X... libraries /usr/X11R6/lib, headers
/usr/X11R6/include
```

Pokud se configure u něčeho zastaví a nedoběhne do konce, musíme problém napravit (obvykle chybí některá knihovna nebo utilita).

Kompilace

Jestliže vše proběhlo, můžeme spustit samotnou kompilaci. Make dokáže také přijímat parametry a jedním z nich je název akce, kterou má dle makefile provádět. Pokud neuvedeme parametr, začne defaultní akce, jenž obvykle znamená kompilaci:

```
make
```

Opět uvidíme řadu kompilačních výstupů a musíme si počkat. Jak dlouho bude překlad trvat závisí především na velikosti projektu, který zpracováváme, a výkonu našeho procesoru.

Když překlad neskončí hlášením pana Errora, máte vyhráno a právě se vám podařilo přeložit první program. Výsledky kompilace jsou ale ještě stále uloženy v adresáři se zdrojovými kódy. Nyní je musíme dostat do systému.

Instalace

Makefile kromě kompilačního postupu obsahuje také kód pro instalaci programu. Takže stačí ze sebe udělat roota (tentorát už bezpodmínečně) a spustit instalaci:

```
su
Password: ********
make install
```

Toto je sice nejjednodušší postup, ale má několik nevýhod. Pokud totiž používáme balíčkovací distribuci (což je dnes skoro každá), nebude při tomto postupu balíčkovací systém o nové aplikaci vůbec vědět. To může mít za následek problém se závislostmi (program v systému doopravdy je, ale podle databáze ne). Navíc se připravíme o možnost komfortní odinstalace a v systému nám tak začne postupem času vznikat nepořádek. Řešením je vytvořit před instalací balíček a ten následně nainstalovat.

Opět to nebude nic těžkého a postačí nám k tomu utilita checkinstall, která dokáže sledovat činnost `make install`, rekonstruovat všechny kroky, posbírat soubory a sestavit balíček.

Místo posledního příkazu tedy použijeme

```
su
Password: ********
checkinstall
```

Program se po krátké přípravě zeptá, pro jakou distribuci má balíček vytvořit (Slackware, RPM nebo Debian), vyžádá si popis programu a po kontrole údajů se již balíček vytvoří a nainstaluje.

Hotovo. Tímto jste zkompilovali a nainstalovali svůj první program.

"Zavuete očíčka pane Čupeuo, tááák. A teď otevuete ... že to nebolelo?" (J. Werich: Až opadá listí z dubu)

Příště se podíváme na nějaké vychytávky a ukážeme si, jak kompilaci vylepšit.

Nebojíme se kompilace - III (ladíme)

Už víme, jak kompilace probíhá, už si umíme sami zkompilovat nějaký prográmek a dnes ukončíme naše povídání tím, že si vysvětlíme, jak udělat kompilaci hezčí, hravější, rychlejší... no prostě lepší.

Skript `configure`, o kterém jsme si již povídali, zkoumá při svém běhu celý systém a snaží se zjistit, co máme a co nemáme. A podle toho přizpůsobuje parametry, které se následně dostanou až ke kompilátoru a ovlivní tak (velmi výrazně) výsledek jeho práce.

Makefile je obyčejný textový soubor, takže jej stačí otevřít v oblíbeném editoru [čti vim]. V makefile se kromě jednotlivých příkazů nacházejí také proměnné. Ty jsou zapsány úplně na začátku a zjednodušují nám změnu celého procesu. Nejzajímavější je pro nás proměnná `CFLAGS`. Ta obsahuje právě parametry kompilátoru a může vypadat třeba takto:

```
CFLAGS= -O3 -march=pentium -ffast-math
```

Existuje několik důvodů, proč bychom je chtěli měnit:

- chceme změnit optimalizaci programu
- chceme překládat na jiném stroji, než na kterém to nakonec poběží
- chceme si prostě hrát

Jen vysvětlení ke druhému bodu: kompilovat můžeme na úplně jiném procesoru nebo dokonce jiné architektuře. Není problém překládat na CPU 486 program pro P4 (i když se to v praxi dělá spíš obráceně).

Parametrů je celá řada a my si popíšeme ty nejzajímavější:

`-mcpu=`

`-march=`

Tyto dva parametry jsou si velmi podobné. Oba optimalizují kód pro zvolený procesor. První z nich však nemění tabulku instrukcí, takže aplikace pak běží i na procesorech nižších generací. Příklad: Pokud použijeme `-mcpu=pentium-mmx`, bude kód skládán tak, aby běžel co nejrychleji na Pentiu MMX, ale nebudou použity žádné instrukce MMX, takže aplikace poběží i na i386. Pokud ale použijeme `-march=pentium-mmx`, budou použity všechny možnosti optimalizace, včetně využití MMX instrukcí, takže výsledek už na ničem nižším nepojede.

Podporované možnosti u procesorů Intel a kompatibilních jsou: *i386, i486, i586, i686, pentium, pentium-mmx, pentiumpro, pentium2, pentium3, pentium4, prescott, nocona, k6, k6-2, k6-3, athlon, athlon-tbird, athlon-4, athlon-xp, athlon-mp, winchip-c6, winchip2 a c3*. Tento výčet není zdaleka konečný a jediný správný. Může se zásadně lišit podle verze gcc a aplikovaných patchů.

`-O`

Určuje míru optimalizace. Tento parametr musí být následován hodnotou (0-3), která určuje, jak moc se kompilátor zabývá vyladěním kódů. Obvykle platí, že čím vyšší stupeň optimalizace zvolíme, tím výkonnější bude výsledný kód a tím delší a časově náročnější bude překlad. Některé aplikace jsou ale na vyšší stupeň citlivé a kompilace neproběhne nebo nakonec aplikace neběží.

`-fexpensive-optimizations`

Aktivuje dodatečně další a náročnější optimalizace.

`-ffast-math`

Při použití tohoto přepínače poruší gcc některá pravidla norem ANSI v zájmu zvýšení výkonu

matematických operací.

`-fforce-mem`
Před prováděním matematických operací převádí matematické operandy do registrů.

`-funroll-loops`
Překladač rozepíše cykly, u nichž je předem znám počet průběhů. Místo "udělej 20x blabla" se tedy ve výsledném kódu objeví přímo "blabla blabla blabla blabla ...". Ve výsledku zvýší velikost výsledného binárního kódu, ale zároveň urychlí jeho běh.

`-fschedule-insns`
Instrukce jsou skládány tak, aby se minimalizovala doba čekání na vstupní data.

To je z nejdůležitějšího asi vše. Chcete-li více, viz `man gcc`.

Otázkou ovšem zůstává, co a kdy optimalizovat. Některé distribuce jako Gentoo nebo SourceMage automaticky kompilují zdrojové balíčky už při jejich instalaci. Takže jsou optimalizovány **všechny** programy i knihovny. V příslušném konfiguračním souboru pak lze nastavit konkrétní kompilační parametry.

To má pochopitelně několik výhod:

- řeší to většinu závislostí
- optimalizuje veškerý kód
- umožňuje udržovat aktuální distribuci
- není nutno čekat na balíčky

Nevýhodou je samozřejmě delší instalace veškerého software (během ní je ovšem možno normálně pracovat). Otázkou všeobjímající kompilace ovšem zůstává, zda se takto ztrávený čas vyplatí. Zarytí Gentooisti budou neobjektivně tvrdit, že rozhodně ano. Zkusme se na celou situaci podívat z nadhledu.

V každé distribuci jsou k dispozici tisíce programů. Stačí si v konzoli dvakrát stisknout tabulátor a hláška `Display all 1637 possibilities? (y or n)` nás jistě přesvědčí, že je tomu skutečně tak. Drtivá většina aplikací však spadá do skupiny systémových utilit, jenž jsou velmi nenáročné na procesorový čas. Jsou to například: ls, grep, less, more, sort, cat, echo, ping a mnoho dalších.

Otázka tedy zní: Vyplatí se nám optimalizovat si i tyto programy? Podle mě ne. Proto je zbytečné volit Gentoo nebo SourceMage jen proto, abychom měli "vyladěný" systém.

Naopak některé důležité programy je vhodné si optimalizovat na míru svému stroji. Jsou to ale spíše náročnější aplikace jako MPlayer, POV-Ray a podobně, kde se může za jistých okolností hodit každé procento. Rovněž na glibc (hlavní systémová knihovna, kterou využívají všechny programy) můžeme ušetřit kousek zatížení procesoru. Při přehrávání mp3 na 3GHz procesoru nás však ani toto nemusí trápit.

Jiná situace samozřejmě nastává v případě, že vlastníte nějaký zvláštní druh hardware (například 64 bitový procesor v serveru). Je pochopitelné, že instalace a použití 32 bitové distribuce "tak, jak je", by zde bylo chybou.

Další stranou mince, kterou je potřeba zvážit, je fakt, že některé distribuce (například Debian) je možno sehnat již předkompilované pro různé druhy platforem, případně můžeme stáhnout upravené balíky (MalýJarda například vyrábí své debianní balíky rovnou pro platformu i686). Máme tak ulehčenou práci, protože optimalizaci už udělal někdo za nás.

Obecně vzato je optimalizace velmi užitečná věc a v případě náročných projektů nebo nově instalovaných programů, které si sami kompilujeme, se nám může hodit. Ovšem i zde platí klasické pořekadlo: "Všeho s Mírou." Tímto uzavíráme náš miniseriálek o kompilacích. Doufám, že se líbil a že jste se dozvěděli něco nového.

Domácí síť - I

Určitě jste se -- stejně jako já -- alespoň jednou dostali do situace, kdy jste zjistili, že počet počítačů v domácnosti značně pokulhává za počtem uživatelů, kteří právě teď potřebují pracovat, případně musí nutně odpovědět na e-mail. Dlouho jsem se bránil myšlence koupit další počítač, ale když jsem zjistil, že cena starého Pentia 200 i s kvalitním monitorem zdaleka nepřesáhne 3000 korun, spočítal jsem si, že tyto investované peníze se mi rychle vrátí. Byl jsem si ovšem vědom závažnější skutečnosti: nevím, jak postavit síť (i když má pouze dva počítače), a už vůbec nevím, jak se to dělá v Linuxu. Dnes už to vím, a protože mě to stálo dost času, chci poradit i čtenářům AbcLinuxu. Proto také prosím odborníky o shovívavost a korekci nepřesností, protože všechno, o čem budu psát, jsem se naučil sám, takže některé věci možná budou nepřesné. Ale výsledek funguje!

Cíl našeho snažení

Hned na začátek tohoto miniseriálu vám chci sdělit, co že to vlastně budeme budovat, k čemu vám to bude a co tím vším získáte. Podle těchto návodů byste měli být schopní postavit si domácí síť běžící na Linuxu. Síť bude mít dva počítače (server, terminál), ale neměl by být problém rozšířit ji o další stanice. Na (každé) stanici bude možné spouštět aplikace ve vašem grafickém prostředí, které je nainstalované na serveru (např. KDE 3.x); problém nebude činit ani tisk. Vytvoříte komfortní pracovní prostředí, které by mělo být pro každého uživatele na kterémkoliv počítači v síti totožné (myšleno pro srovnání strojů, ne uživatelů).

Než začneme

Co všechno musíte koupit...

Na začátek si řekněme, co je k vytvoření domácí sítě (síťky) potřeba. Vycházím pochopitelně ze své situace, ale pokusím se abstrahovat. Síť postavíme na bázi klient-server. Na serveru poběží všechny procesy, které budou poskytovány klientovi na žádost. Na klientovi spustíme pouze procesy nutné pro jeho správu a pro síťové spojení. Co všechno tedy potřebujeme?

- server: bohatě stačí počítač s frekvencí kolem 400 MHz, množství operační paměti RAM by mělo rozhodně přesáhnout 64 MB (čím více, tím lépe); důležitý je rychlý disk (7200 otáček), ale není podmínkou;
- stanice: jakýkoliv počítač (pochopitelně pomalejší než server), měl by mít alespoň 16 MB RAM (nutné minimum pro grafické prostředí);
- dvě síťové karty a síťový kabel;
- operační systém Linux (na serveru už nainstalovaný); instalační diskety (popis jejich vytvoření najdete na instalačním cd); není nutné mít na serveru i stanici stejnou distribuci/verzi Linuxu; v tomto článku vycházím ze svých zkušeností s distribucí Slackware 8.0 a 9.0
- a pochopitelně čas, trpělivost a ochotu se něco nového naučit -- tohle je opravdu nutný vklad, ale berte to pozitivně

...a kolik to stojí?

Předpokládám, že server máte; snad i klienta. Pokud máte jen jeden počítač a koupíte druhý, jako server by měl sloužit ten rychlejší. (Ale to je snad jasné...) Pokud budete pořizovat klienta, pak vězte (kdo se neorientujete), že zmíněné Pentium 200 v konfiguraci 32 MB RAM/1.2 GB harddisk/disketová mechanika/grafická karta 1--4MB RAM stojí cca 1500 Kč. Součástí asi nebude CD-ROM, ale to nevadí, naopak si myslím, že je zbytečná. Kvalitní digitální monitor se značkovou elektronikou ("vnitřkem") přijde na cca 1500 Kč. Síťovou kartu koupíte v bazaru za 50--100 korun, jedná se o "krabicové" zboží (jsou naházené v krabici někde v rohu a obsluhu otravuje už to, že vám kvůli tomu musí vypsat paragon). Může se stát, že síťovka nebude funkční, proto je vhodné koupit raději novou (cca 200 Kč/kus); záleží na vás. Relativně velký problém nastává při koupi kabelu. V prodejnách VELICE neradi stříhají kabel na míru, protože celá práce neodpovídá ceně kabelu (10 Kč/m); proto počítejte s tím, že největší položkou v ceně kabelu bude práce. Není vhodné natahovat kabel venkovním prostředím, hrozí nejen zničení počítačů, ale dokonce i vyhoření domu/bytu (blesk si vybírá to nejlepší místo). Také není vhodné instalovat kabel delší než 50 metrů -- není pro to určen (elektromagnetické pole, ztrátovost při přenosu). Optimální délka kabelu by tedy neměla přesáhnout 20 metrů (z místnosti do místnosti podél zdi to vyjde na cca 10--15 metrů).

Velice důležitým faktorem je typ kabelu. Požadujte **křížený** kabel pro 10/100 MBit ethernet. Je to velmi důležité (jeden z vodičů -- bílý? -- kříží na jednom konci ostatní). Pokud si tímto nejste jisti, z problémů se nedostanete. Cena kabelu by neměla přesáhnout 200-300 korun. Celkové náklady se tedy pohybují kolem 3000-3500 Kč (nepočítám v to cenu operačního systému, který je zadarmo.

Táhneme!

První zásadní věcí je pochopitelně natažení kabelu po bytě. Zmiňuji se o tom pouze proto, že vám to zabere určitou dobu. Kabel veďte tak, aby nebyl na žádném místě "skřípnutý" (např. pod prahem dveří), neměl by být ohnutý násilím. Do zdířky síťové karty by měl být zacvaknutý pod přirozeným úhlem. Všechny takové kritické body jsou potenciálním zdrojem nepříjemností.

Začněme Stanicí

Pojmenujme si pro naše účely oba počítače jmény Server a Stanice. Nainstalujte síťovou kartu (fyzicky) do počítače a ujistěte se v setupu, že používá jedinečné IRQ a adresu. Do setupu se dostanete při startu počítače v okamžiku, kdy se "počítá" paměť nejčastěji stiskem klávesy `Del` nebo podobné (měla by být zobrazena na monitoru). U starších počítačů (se kterými pracujeme) nemáte možnost nastavovat takové "citlivé" hodnoty, stačí když víte, která IRQ jsou volná. Pokud dvě IRQ kolidují, počítač stoprocentně "tuhne", a to nejen ve Windows. Na tomto místě je nutné poznamenat, že síťová karta má vlastní paměť, kde tyto informace uchovává. Lze je nastavit programem, který získáte na webových stránkách výrobce. Tyto utility běží obvykle v DOSu, pokud nemáte starý MSDOS, použijte kupř. DR-DOS. Vytvořte si bootovací disketu spolu s uvedenou rutinou a kartu řádně nastavte (na hodnoty, které si poznačíte). Jako způsob spojení nastavte 10/100Mbitový ethernet. Obvykle se není vhodné zadávat zde způsob konfigurace PlugAndPlay, nastavte hodnoty "natvrdo". Tuto fázi se nevyplácí podceňovat, měli byste ji provést každopádně.

Pokud jste kartu nastavili a jste přesvědčeni, že nebude kolidovat s jinými zařízeními, je hardwarová příprava Stanice u konce. Zasuňte konektor kabelu do síťové karty a přesuňte se k Serveru.

Na řadě je Server

Tvrdé zboží (hardware)

Nainstalujte na Serveru síťovou kartu stejným způsobem jako na Stanici. Spusťte Linux. Sledujte výpisy jádra, jestli se objeví (přibližný) název síťové karty. Pokud ano (měl by), je vše v pořádku (alespoň z 50%). Přihlaste se jako root a zadejte

```
dmesg | less
```

Tento příkaz vám znovu zobrazí informace jádra, které se produkují při startu systému. Zkontrolujte, zda údaje o síťové kartě (IRQ, adresa) odpovídají těm údajům, které jste nastavili dosovou utilitou. Výpis může vypadat např. takto:

```
...
PCI: Found IRQ 11 for device 00:03.0
PCI: Sharing IRQ 11 with 00:07.2
3c59x: Donald Becker and others. www.scyld.com/network/vortex.html
00:03.0: 3Com PCI 3c590 Vortex 10Mbps at 0x1040. Vers LK1.1.16
00:03.0: Overriding PCI latency timer (CFLT) setting of 64, new
value is 248.
...
```

Kromě stránky, kde lze najít novou verzi ovladačů této síťové karty, jsme se dozvěděli, že se jedná o kartu 3com, model 3c590, která má IRQ nastavené na 11, adresu na 0x1040 hexadecimálně a způsob komunikace na 10Mbps. Modul pro ovládání karty se tedy v pořádku načetl.

Pokud ve výpisu informace o kartě nenajdete, je nutné načíst modul ručně. Pokud znáte název karty nebo chipsetu, zadejte:

`modprobe -l | grep` *`nazev_karty_nebo_chipsetu`*

Pokuste se odhadnout nebo určit, který modul je ten pravý. Vložte jej příkazem `modprobe` *`nazev_modulu`*. Pokud na vás nevystřelí množství chybových hlášení, našli jste jej; přesvědčte se příkazem `lsmod`. Pokud se chybová hlášení objeví, nastává čas natáhnout Google a hledat. Je to velice nepravděpodobné, proto hodně štěstí

Měkké zboží (software)

Modul je načten, nastává čas nastavit síťové služby. Způsob nastavení se v každé distribuci nepatrně liší. Ať už to provedete prostřednictvím nějaké utility, nebo ručně, výsledek musí být stejný. Proto přímo popíšu nastavení konfiguračních souborů; jejich umístění v adresářové struktuře se v jednotlivých distribucích může lišit. Předpokládám, že váš systém přišel na svět standardní instalací se síťovou podporou, proto zde tento proces neřeším.

První věcí, kterou nastavíme, jsou síťové adresy Serveru a Stanice. Existuje způsob, jak adresu stanicím přidělovat dynamicky, ale v případě jedné stanice to nemá význam. Adresy všech stanic v síti (tedy i Serveru!) jsou uloženy v souboru `/etc/hosts`:

```
# hosts # This file describes a number of hostname-to-address
# mappings for the TCP/IP subsystem. It is mostly
# used at boot time, when no name servers are running.
# On small systems, this file can be used instead of a
# "named" name server. Just add the names, addresses
# and any aliases to this file...
```

```
#
# By the way, Arnt Gulbrandsen
```

Tento soubor, zde konkrétně pocházející z mého Slackwaru 9.0, poskytuje základní informace o tom, jak zadávat údaje do souboru. Dočteme se, že na malých systémech je tento soubor vhodnější než nameserver (který v podstatě totéž dělá na velkých systémech). Je také nevhodné pojmenovávat loopback adresu jménem počítače. (Tento pomyslný klient slouží k testování funkčnosti sítě; počítač vlastně komunikuje sám se sebou, ale přes síťovou kartu.) Formát údajů je následující:

```
sitova_adresa nazev_pocitace.nazev_site aliasy
```

Položka `localhost` s adresou `127.0.0.1` je "téměř" povinná. Setkal jsem se názorem, že by měla být uvedena jako první. Něco o tom, jakou adresu vybrat, se dočtete v Networking HOWTO. Adresní prostor pro soukromé sítě nepřipojené do Internetu začíná právě na hodnotě 192.168.1.0.

Nyní přiřadíme síťovou adresu vlastní kartě. Ve Slackwaru se jedná o soubor `/etc/rc.d/rc.inet1` (výřez):

```
IPADDR="192.168.1.10" # REPLACE with YOUR IP address!
NETMASK="255.255.255.0" # REPLACE with YOUR netmask!
GATEWAY="" #REPLACE with YOUR gateway!
BROADCAST=`/bin/ipmask $NETMASK $IPADDR | cut -f 1 -d ' '`

/sbin/ifconfig lo 127.0.0.1
/sbin/route add -net 127.0.0.0 netmask 255.0.0.0 lo
/sbin/ifconfig eth0 ${IPADDR} broadcast ${BROADCAST} netmask
${NETMASK}
/sbin/route add -net 127.0.0.0 netmask 255.0.0.0 lo
```

Tento soubor používá z důvodu flexibility proměnné, nastavení ale můžeme provést pomocí přímých hodnot. Bránu (gateway) není nutné nastavovat. K podrobnostem vizte výše zmíněné HOWTO. Kontrolu nastavení provedeme příkazem `ifconfig`. Výpis by měl vypadat nějak takto:

```
/etc/rc.d: ifconfig
eth0  Link encap:Ethernet HWaddr 00:A0:24:CC:A7:34
      inet addr:192.168.1.10 Bcast:192.168.1.255
Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:8143 errors:0 dropped:0 overruns:0 frame:0
      TX packets:7321 errors:0 dropped:0 overruns:0 carrier:0
      collisions:118 txqueuelen:100
      RX bytes:4289868 (4.0 Mb) TX bytes:2572950 (2.4 Mb)
      Interrupt:11 Base address:0x1040

lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:80 errors:0 dropped:0 overruns:0 frame:0
      TX packets:80 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:5632 (5.5 Kb) TX bytes:5632 (5.5 Kb)
```

Další fází je spuštění požadovaných služeb. Opět zde platí, že každá distribuce poskytuje jiné nástroje pro jejich nastavení. Zásadní program se jmenuje portmapper a spouští se (ve startovacích

skriptech v adresáři `/etc/rc.d/`) příkazem `rpc.portmap`. Zprostředkovává základní komunikaci mezi počítači. Pokud neběží, spusťte jej a přesvědčte se příkazem `rpcinfo`, výpis by měl být následující:

```
rpcinfo -p

program verz proto port
100000    2   tcp   111 portmapper
100000    2   udp   111 portmapper
```

Tento díl ukončíme testem, zda je síťová komunikace na serveru funkční. Zadejte příkaz `ping server` a sledujte výpis (musíte jej sami ukončit klávesami Ctrl-C). Pokud vypadá nějak takto,

```
ping server

PING server.byt (192.168.1.10): 56 octets data

--- server.byt ping statistics ---
324 packets transmitted, 0 packets received, 100% packet loss
```

tak není síť nastavená správně. Hledejte v archívu diskuzí na AbcLinuxu, je to častý dotaz. Bezchybný výpis je např. ten následující:

```
ping server

PING server.byt (192.168.1.10): 56 octets data
64 octets from 192.168.1.10: icmp_seq=0 ttl=64 time=0.3 ms
64 octets from 192.168.1.10: icmp_seq=1 ttl=64 time=0.1 ms
64 octets from 192.168.1.10: icmp_seq=2 ttl=64 time=0.1 ms

--- server.byt ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.3 ms
```

Pokud všechno funguje, máme Server téměř připravený pro síťovou instalaci Linuxu na Stanici. O té bude pojednávat druhý díl našeho vašeho seriálu. Zatím uvažujte o tom, jakou distribuci nainstalujete na Stanici. Už máte kabel?

Domácí síť - II

Dokončení přípravy Serveru

Pokud vám funguje síťová komunikace na Serveru, je systém připraven ke komunikaci s jiným počítačem. Teď je ale nutné specifikovat typ komunikace (protokol a služby) a nastavit sdílené zdroje (adresářový strom s balíčky distribuce, kterou chceme instalovat na Stanici).

Network File System -- NFS

NFS je systém určený k **nezabezpečenému** sdílení zdrojů (souborů). Na straně serveru poskytneme adresáře ke sdílení (export adresářů), které na klientovi připojíme (mount) do existující adresářové struktury. Exportování adresářů umožňuje program (daemon, démon) `nfsd`. Ten se spouští

skriptem z adresáře /etc/rc.d/ (ve Slackware je to /etc/rc.d/rc.nfsd, v jiných distribucích pravděpodobně podobný skript nebo prostřednictvím rutiny typu setup). Spustit jej pochopitelně musí root, nejlepší je nastavit jeho spouštení po startu systému. To, jestli démon běží, zjistíte příkazem

```
ps aux | grep nfsd
root    164  0.0  0.0    0    0 ?   SW   17:06    0:00 [nfsd]
```

Pokud bude výstup z programu ps prázdný, program neběží. Nyní je nutné specifikovat adresáře, které mají být poskytnuty "do sítě". Je vhodné poskytnout je pouze konkrétním/u počítači/počítačům. Editujte soubor /etc/exports (bude pravděpodobně prázdný) a dopište do něj adresář, ve kterém se nacházejí zdrojové balíčky distribuce. Formát každého řádku je následující:

```
adresar     IP_adresa_nebo_jmeno_klienta(priznaky)
```

Příznaky specifikují přístup k adresáři ze strany klienta. Pro podrobný popis vyhledejte man exports. První z níže použitých znamená, že klient může obsah adresáře pouze číst, a to i když má adresář nastavená práva zápisu pro všechny uživatele. Předpokládejme, že distribuci máte připravenu v adresáři /install (volitelně přímo z připojeného cédéčka).

```
# See exports(5) for a description.
# This file contains a list of all directories exported to other
computers.
# It is used by rpc.nfsd and rpc.mountd.


/install     stanice.byt(ro,insecure)  # nebo cd-rom?
#/mnt/cdrom  stanice.byt(ro,insecure)
```

Pokud démon nfsd běží, restartujte jej; nejčastěji příkazem /etc/rc.d/rc.nfsd restart nebo podobným. Tím dojde k aktualizaci seznamu exportovaných adresářů. Pokud neběží, spusťte jej. Tento démon spustí v případě požadavku na připojení adresáře /install rutinu rpc.mountd, která komunikuje se stanicí a poskytne na ní běžícímu nfsd "odkaz" (resp. "obraz") adresáře /install. Ten je na stanici připojen do adresářové struktury. Připojení tedy funguje stejně jako v případě lokálního média. Úspěch operace ověřte (jako root) příkazem exportfs:

```
exportfs
/install    stanice.byt
```

Pro jistotu

Některé distribuce mohou implicitně zakazovat přístup k systémovým službám. Proto se přesvědčte, že v souboru /etc/hosts.allow není žádný záznam, nebo je tam záznam

```
ALL:ALL
```

Je ovšem nebezpečné (a hloupé) používat takový soubor v "otevřené" síti, proto jej po instalaci změňte (viz man portmap). Analogicky: soubor /etc/hosts.deny by měl být prázdný -- nejsou žádní klienti, jimž by byl zakázán přístup k síťovým službám (opět: později vhodné změnit).

Příprava instalace

Nyní je tedy vše připraveno, zbývá ještě vytvořit bootovací a instalační disketu/diskety. To se v každé distribuci liší, popis najdete na instalačním CD. Pochopitelně je nutné použít diskety určené pro síťovou instalaci. Obvykle jejich počet nepřesáhne tři kusy. Po jejich vytvoření se ujistěte, že jsou k dispozici zdrojové balíčky distribuce, síťová karta je nainstalovaná, běží portmapper (předchozí díl) a démon `nfsd` čeká na síťové spojení. Pusťte k počítači přítelkyni/manželku/syna/dceru (nehodící se škrtněte) a doufejte, že je nebudete muset rušit. Vy se přesuňte na (delší dobu) ke Stanici.

Instalace systému na Stanici

Nastavení síťové komunikace

Do disketové mechaniky vložte startovací disketu a propracujte se k síťové instalaci (bude nutné vložit další disketu s podporou sítě). Během tohoto procesu by měl instalační program najít a nakonfigurovat síťovou kartu. Pokud se tak nestane, citelně "odstraňte" osobu sedící u Serveru a položte dotaz do diskuze na AbcLinuxu. Je totiž pravděpodobné, že je karta na Stanici příliš stará/nová a distribuce, kterou jste vybrali, ji nepodporuje. Je to nepravděpodobné, ale stát se to může. Řešením je zvolit jinou verzi nebo distribuci.

Pokud je karta správně nastavena, program se vás zeptá na zdroj instalačních balíčků -- vyberte NFS (další volby bývají HTTP nebo FTP). V následujících dialogových oknech zadejte údaje o síťovém spojení. Na tomto místě musím maximálně abstrahovat, takže se omezím na přehlednou tabulku.

DHCP	(ne)
Adresa vašeho stroje	192.168.1.20
Síťová maska/Netmask	255.255.255.0
Gateway	(žádná)
Adresa serveru, ze kterého instalujete	192.168.1.10
Adresář, ve kterém jsou na serveru uloženy balíčky	`/install`

Tato část je kritická. Pokud něco nefunguje, projeví se to právě nyní. Nejčastější chyby, které mohou ovlivnit instalaci, jsou: na serveru neběží některá služba (portmapper, nfsd), špatně vyexportovaný adresář, špatně zadaná cesta k adresáři (v adresáři `/install` na serveru by měl být kořenový adresář instalačního CD), zakázaný přístup ke službě, nefunkční síťová karta na Stanici.

Vlastní proces instalace

Dostali jsme se k meritu věci. Pokud se spustí instalační program v celé své kráse (obvykle grafický), máte vyhráno. Vyberte **pouze** balíčky, které jsou nutné pro běh Stanice, její zasíťování a běh x-serveru. Nemá smysl instalovat grafická prostředí nebo manažery, multimediální programy, servery (kromě nfsd), podporu SCSI (pokud na Stanici není harddisk tohoto typu, což je velice nepravděpodobné apod. Kompletní instalace by měla být naprosto minimální, řádově pod 100 MB, ale může to být i mnohem méně. Pro x-server nainstalujte pouze základní sadu fontů, neinstalujte freetype fonty; ty budeme sdílet ze Serveru. Vhodnými doplňkovými programy jsou Midnight Commander a mpg123/mpg321 pro přehrávání mp3 z příkazové řádky. Ostatní programy nemají

opodstatnění. Dokončete instalaci zadáním názvu Stanice (tedy stanice), domény (byt) a ujistěte se, že se při každém startu načtou ovladače síťové karty a spustí se síťové služby (portmapper a nfsd). Vytvořte si startovací disketu pro případ, že se LILO nebo jiný bootovací manažer (GRUB) nenainstaluje správně. Pokud instalátor nabízí nastavení x-serveru, nebraňte se; udělali byste to takjakotak později. Instalace končí restartem systému.

Tenkrát poprvé...

Při nabíhání systému sledujte výpisy jádra, zda se nastaví síťová karta a síťové rozhraní. Pokud ano, všechno důležité je hotovo. Pokud ne, prostudujte výpis jádra (`dmesg | less`). Optimální výpis vypadá např. takto (výřez):

```
...
via-rhine.c:v1.10-LK1.1.14 May-3-2002 Written by Donald Becker
http://www.scyld.com/network/via-rhine.html
PCI: Found IRQ 11 for device 00:12.0
PCI: Sharing IRQ 11 with 00:10.0
eth0: VIA VT6102 Rhine-II at 0xec00, 00:0a:e6:61:48:d9, IRQ 11.
eth0: MII PHY found at address 1, status 0x7849 advertising 05e1
Link 0000.
...
Installing knfsd (copyright (C) 1996 okir@monad.swb.de).
...
```

Ověřte, že běží portmapper (viz výše). Příkazem

```
ifconfig
```

```
eth0   Link encap:Ethernet HWaddr 00:0A:E6:61:48:D9
       inet addr:192.168.1.20 Bcast:192.168.1.255
Mask:255.255.255.0
       UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
       RX packets:299767 errors:0 dropped:0 overruns:0 frame:0
       TX packets:313747 errors:0 dropped:0 overruns:0 carrier:0
       collisions:29616 txqueuelen:100
       RX bytes:42149005 (40.1 Mb) TX bytes:103680242 (98.8 Mb)
       Interrupt:11 Base address:0xec00

lo     Link encap:Local Loopback
       inet addr:127.0.0.1 Mask:255.0.0.0
       UP LOOPBACK RUNNING MTU:16436 Metric:1
       RX packets:86 errors:0 dropped:0 overruns:0 frame:0
       TX packets:86 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:0
       RX bytes:6092 (5.9 Kb) TX bytes:6092 (5.9 Kb)
```

ověříte, že je síťové rozhraní funkční. Příkazem

```
route -n
```

```
Kernel IP routing table
Destination   Gateway   Genmask        Flags Metric Ref Use Iface
192.168.1.0   0.0.0.0   255.255.255.0   U      0     0   0   eth0
127.0.0.0     0.0.0.0   255.0.0.0       U      0     0   0   lo
```

si ověříte, že je správná směřovací tabulka síťových paketů, tzn., že data posílaná přes síťovou kartu `eth0` mají kam směřovat.

Nyní zkuste

```
ping server

PING server.byt (192.168.1.10): 56 octets data
64 octets from 192.168.1.10: icmp_seq=0 ttl=64 time=0.1 ms
64 octets from 192.168.1.10: icmp_seq=1 ttl=64 time=0.0 ms
64 octets from 192.168.1.10: icmp_seq=2 ttl=64 time=0.0 ms

--- server.byt ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.0/0.0/0.1 ms
```

a dozvíte se, že Server odpovídá. Heureka! Pokud je odezva následující

```
ping server

PING server.byt (192.168.1.10): 56 octets data

--- server.byt ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
```

portmapper na straně Serveru neodpovídá. Chyba "Network unreachable" znamená, že portmapper neběží na stanici. Další hlášení mohou značit špatnou konfiguraci síťové karty. To vše ale lze vyřešit, protože jste přece už nainstalovali systém. Hledejte opět v archivu diskuzí na AbcLinuxu.

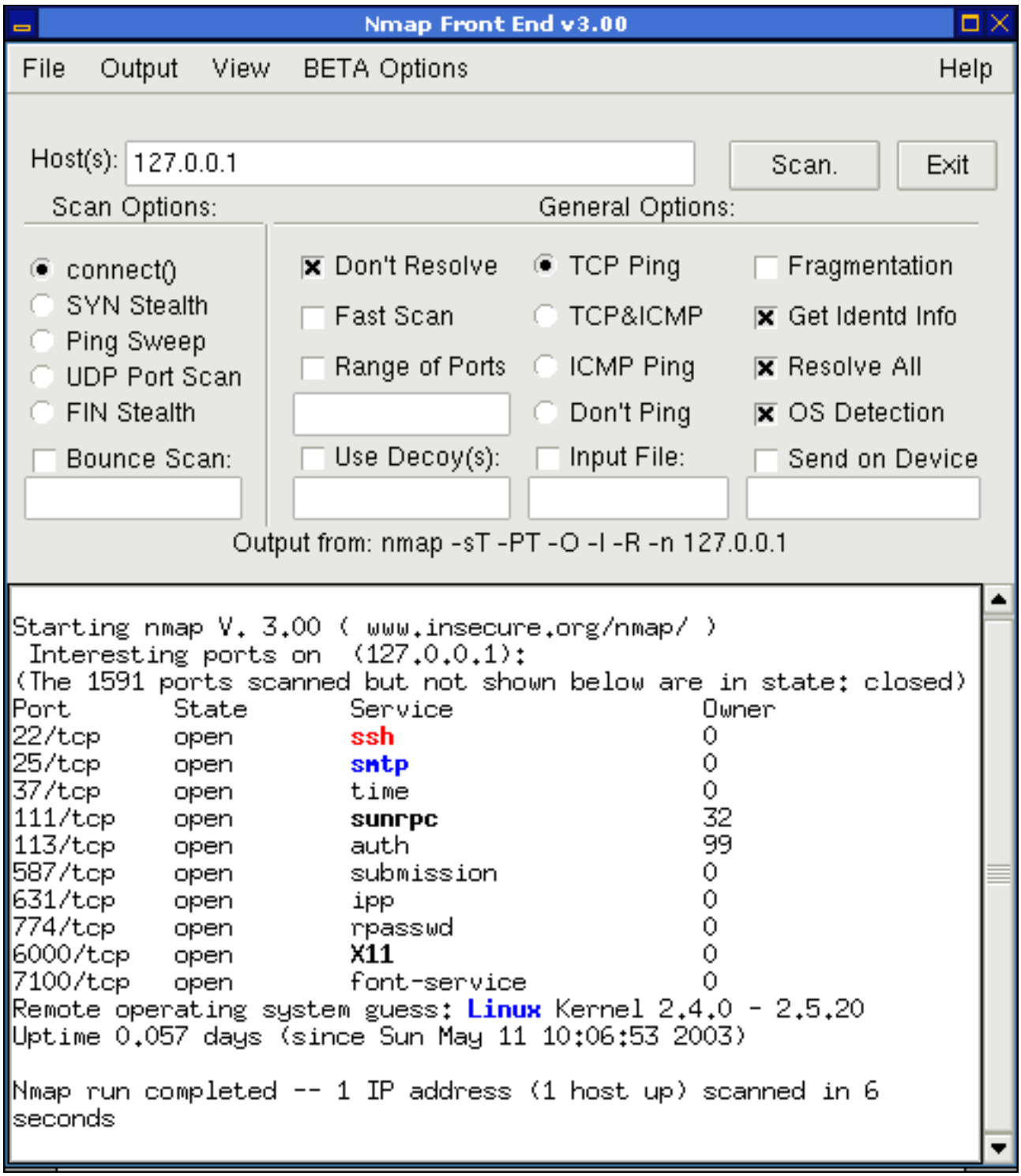
Pokud se vám síťová instalace nezdaří na první pokus, nezoufejte, existují lidé, kteří to dělali i desetkrát (třeba já; chyba byla v síťové kartě). Příště vytvoříme ze Stanice grafického klienta, na kterém budete moci pracovat stejně komfortně jako na Serveru. Budu vděčný za jakékoliv připomínky či dotazy nebo kritiku.

Domácí síť - III

V minulém díle jsme navázali spojení mezi Serverem a Stanicí. Dnes instalaci dokončíme a proměníme Stanici v grafický terminál. Tato část je poněkud namáhavá na čas, protože "vyladění" provozu vyžaduje jistou trpělivost při zkoušení mnoha nastavení. Ale je možné, že vám se to povede rychleji než mně.

Ještě k NFS a bezpečnosti

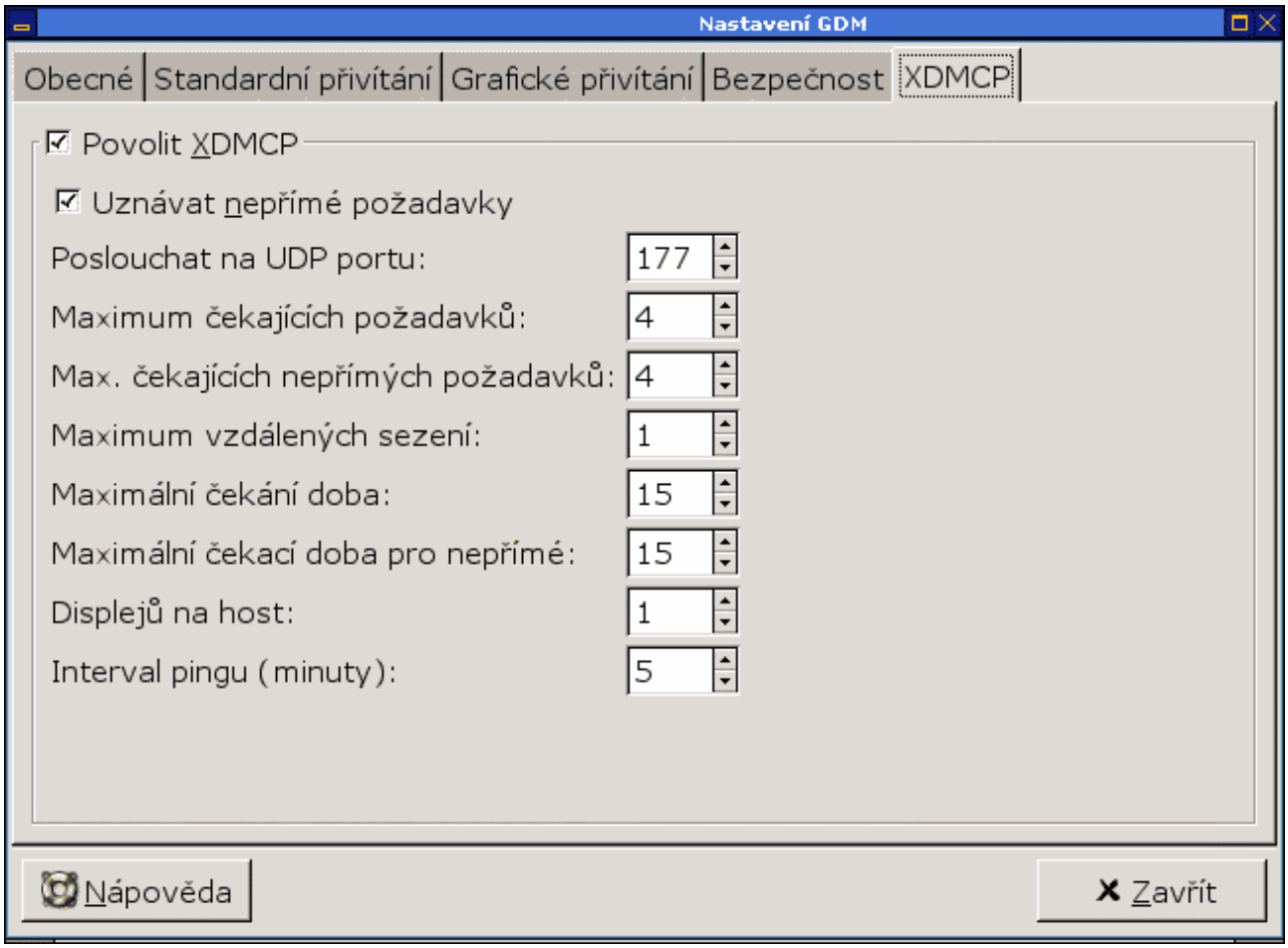
Kromě toho, že budete na Stanici pracovat v grafickém prostředí, můžete pochopitelně využívat i další konzole systému. Tam oceníte Midnight Commander a, pokud máte na Stanici zvukovou kartu, můžete také poslouchat hudbu v mp3 (program mpg123). Máte tedy možnost přimountovat si adresáře s hudbou do adresářové struktury Stanice. (O zvuku vizte níže.) Pro případ, že se potřebujete z konzole dostat do adresářové struktury na Serveru, oceníte program ssh (a pochopitelně server sshd běžící na obou počítačích). Je nevhodné spouštět na serveru služby telnet nebo ftp, protože tím nabízíte obsah harddisku celému světu. Zde je na místě alespoň krátce se zmínit o bezpečnosti. Základním principem je: aktivovat pouze služby/servery, které potřebujete. To určitě **není** ftp a telnet. Zamezte tedy jejich spuštění při startu systému. Dále je vhodné nakonfigurovat soubory `/etc/inetd.conf`, `hosts.deny` a další, nemluvě o instalaci firewallu. Pro základní informační přehled o tom, co "nabízíte" všem zájemcům, spusťte jako root příkaz `nmapfe`.



Jak vidíte, na můj počítač můžete poslat své dokumenty k tisku (tiskový systém CUPS na portu 631), můžete se pokusit připojit pomocí šifrovaného spojení (ssh), odesílat přes mě své maily (smtp), připojit se k mému x-window systému a nebo používat mé freetype fonty. Tohle není pozvánka! Naštěstí jsem modemový uživatel, kterému přiděluje IP adresu provider náhodně, ale existují automaty, které testují i takové uživatele. Už jsem měl několik návštěv přes ftp. Bezpečnost je citlivé téma, které se nevyplatí podceňovat! (Pro mě je to oblast, na které budu pro svůj klid pracovat v nejbližší budoucnosti.)

Konečně login?

Pojďme k hlavnímu cíli. Jak donutit grafický přihlašovací manažer, aby naslouchal požadavkům ze sítě (Stanice), příp. aby požadavky vysílal (Server)? V Linuxu se používá standardní xdm. Obě majoritní prostředí (GNOME a KDE) nabízejí své manažery (kdm a gdm), které jsou velmi přívětivé. Jejich konfigurace pro naše účely je téměř nulová; musíme pouze nastavit, aby povolily protokol XDMCP, tj. aby přijímaly požadavky na připojení ze sítě. Nezáleží na tom, který z nich používáte; vyberte si ten, který se vám více líbí. Jejich základní konfiguraci si krátce probereme níže.



Stanice

Předpokládejme, že na Stanici máte nainstalované pouze XFree se standardními součástmi, mezi které patří i xdm. Soubor `/etc/X11/xdm/xdm-config` musí obsahovat (mimo jiné!) následující řádek (obvykle bývá zakomentovaný "vykřičníkem" a je uveden jako poslední):

```
DisplayManager.requestPort: 0
```

Tím nastavíme systém x-window tak, aby naslouchal XDMCP nabídkám k připojení.

Server

xdm

Tento správce přihlášení má celkem významnou nevýhodu: nezobrazuje možná sezení, která si můžete vybrat. Spustí tedy to, které je nastaveno jako standardní (`/etc/X11/xinit/xinitrc`). Pro správnou komunikaci proveďte následující nastavení:

Soubor `/etc/X11/xdm/Xaccess` musí obsahovat řádek

- `CHOOSER BROADCAST   #any indirect host can get a chooser`

Česky řečeno: nabízej možnost grafického připojení **všem**, kdo projeví zájem. Jak jistě chápete, hrozí tu možnost zneužití či útoku. Místo hvězdičky je tedy vhodné zadat IP adresu počítače, který se smí připojit. Pokud by toto nastavení nefungovalo, použijte pouze "hvězdičku" příp. IP adresu.

kdm

Pro konfigurační soubor kdm (`/opt/kde/share/config/kdm/Xaccess`) platí totéž, co pro výše uvedený konfigurační soubor xdm:

```
*      CHOOSER BROADCAST
#*        # NEBO hvězdička (všichni) pro případ, že předchozí
nefuguje
```

Ujistěte se, že v souboru `/opt/kde/share/config/kdm/kdmrc` je v sekci `[XDMCP]` položka `enable=true`.

gdm

V souboru `/etc/X11/gdm/gdm.conf` je v sekci `[XDMCP]` položka `enable=true`. Narozdíl od předchozích dvou správců neřeší gdm bezpečnost speciálním souborem `Xaccess`, ale prostřednictvím systémových souborů `/etc/hosts.allow` a `/etc/hosts.deny`. V nich je vhodné a nutné! specifikovat klienty, kteří se mohou připojit. Ale nejen to, je také vhodné realizovat zde základní bezpečnostní opatření (opravdu jen základní!). Např.

```
ALL:ALL EXCEPT LOCAL, .byt
```

v souboru `/etc/hosts.deny` zakáže přístup ke všem službám všem klientům kromě počítačů z domény `.byt`. Pokud ale znovu spustíte program `nmapfe`, zjistíte, že se mnoho nezměnilo -- počítač je pořád "dokořán", protože doméně .byt nabízíme všechny služby. Ale alespoň víte, že tady jste udělali maximum. Znovu připomínám: bezpečnost se zajišťuje jinými způsoby.

Konečně login!

Pokud na serveru běží x-window systém (a také některý ze zmíněných správců), pokusíme se ze Stanice připojit. Pro testovací účely použijme příkaz `X -broadcast`. Spustí se XFree, hledá v síti pomocí XDMCP server, ke kterému by se mohl přihlásit. Pokud jej najde, zobrazí jeho okno přihlášení. Pokud jej nenajde, spustí se "čisté" XFree s křížovým kurzorem. Standardně neprobíhá žádné bezpečnostní ověření! Pokud nic nenajde, žádný server nevysílá do sítě (bude nutno

restartovat manažer přihlášení na Serveru), nebo klient není autorizován. Pokud se zobrazí přihlašovací okno se jmény uživatelů na Serveru (kdm, gdm), máte vyhráno. Ukončete server klávesovou kombinací `Ctrl-Alt-Backspace`. Nyní je vhodné změnit na Stanici runlevel číslo 4 (grafické režim) tak, aby se po startu systému automaticky hledaly počítače v síti, ke kterým je možné se připojit a zobrazilo se přihlašovací okno. Soubor `/etc/rc.d/rc.4` obsahuje tyto příkazy:

```
#!/bin/sh
#
# rc.4      This file is executed by init(8) when the system is
being
#           initialized for run level 4 (XDM)
#
# Version:    @(#)/etc/rc.d/rc.4 2.00 02/17/93
#
# Author:     Fred N. van Kempen,
```

Tím tedy získáte možnost připojit se ke všem počítačům, které připojení v daném segmentu sítě nabízejí. Připojení ke konkrétnímu počítači (třeba i do internetu) se realizuje parametrem `-query IP_adresa_pocitace`.

Písma

Ze Serveru...

Předpokládáme, že na Stanici nejsou nainstalovány fonty pro X-window (kromě základních 100dpi- a 75dpi-fontů). Zpřístupníme tedy freetype/truetype fonty ze Serveru. Podmínkou toho je, že na serveru běží X Font Server (xfs). Ten je schopen exportovat katalogy fontů "do sítě"; na Stanici běží XFree (nejlépe verze 4.3.0), které je schopno s těmito písmy pracovat. Ověřme si na Serveru, že xfs běží:

```
ps aux | grep xfs

root     260  0.0  1.1  7128 5748 ? S    10:07 0:01
/usr/X11R6/bin/xfs -daemon
```

X font server je někdy nutné také nakonfigurovat. (Pozn. pro uživatele KDE: "Instalátor písem" v Ovládacím centru KDE to za vás neudělá! Ten pouze nainstaluje písma do prostředí KDE, ale už ne do samotného XFree.) Protože je konfigurační soubor trochu delší, přikládám jej jako přílohu (jedná se o soubor `/etc/X11/fs/config`). V každém adresáři, který zařazujeme do katalogu písem, musí existovat soubory `fonts.dir` a `fonts.scale`. O jejich vytvoření vizte články o truetype fontech v Mozille; je pravděpodobné, že je vše v pořádku.

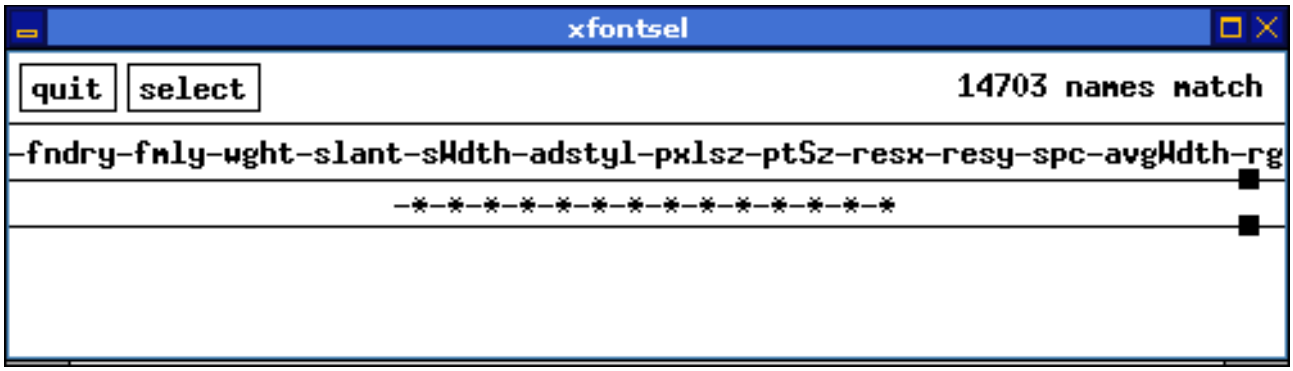
...na Stanici

Na Serveru tedy běží xfs a poskytuje katalog písem. Na Stanici musí být XFree nastaveno tak, aby dokázalo tento seznam adresářů použít ve svých vlastních katalozích písem. Zde už xfs běžet nemusí; pouze nastavíme několik parametrů v souboru `/etc/X11/XF86Config` (jedná se o

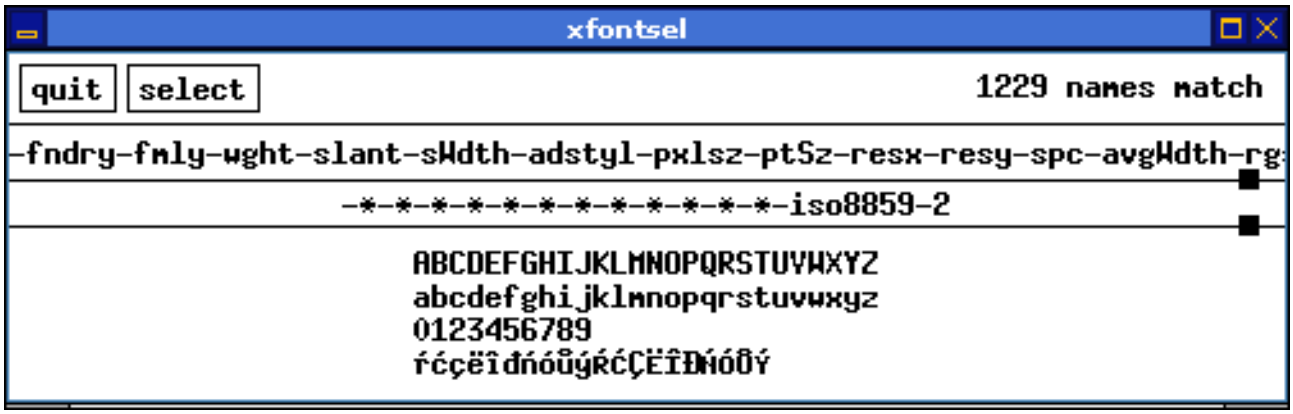
případ XFree 4.3.0!). Do cesty písem (FontPath) přidáme adresu počítače, na kterém běží font server (192.168.1.10), a specifikujeme port, na kterém naslouchá (obvykle 7100). Více `man xfs`. Nesmíme zapomenout také na načtení modulů, které umožňují práci s těmito fonty (sekce "Module" a názvy "freetype", "xtt" a další).

```
Section "Files"
   RgbPath    "/usr/X11R6/lib/X11/rgb"
   ModulePath "/usr/X11R6/lib/modules"
   FontPath   "tcp/192.168.1.10:7100"
   FontPath   "/usr/X11R6/lib/X11/fonts/misc/"
   FontPath   "/usr/X11R6/lib/X11/fonts/75dpi/"
   FontPath   "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection
Section "Module"
   Load "record"
   Load "extmod"
   Load "dbe"
   Load "dri"
   Load "glx"
   Load "xtrap"
   Load "xtt"
   Load "freetype"
   Load "type1"
   Load "speedo"
EndSection
...
```

Další informace o font serveru získáte příkazy `xfsinfo`, `fslsfonts`, `showfont`, `xlsfonts` a pochopitelně `xfontsel`. Následuje ukázka tohoto programu sejmutá na Stanici, tedy v "odlehčeném" x-systému.

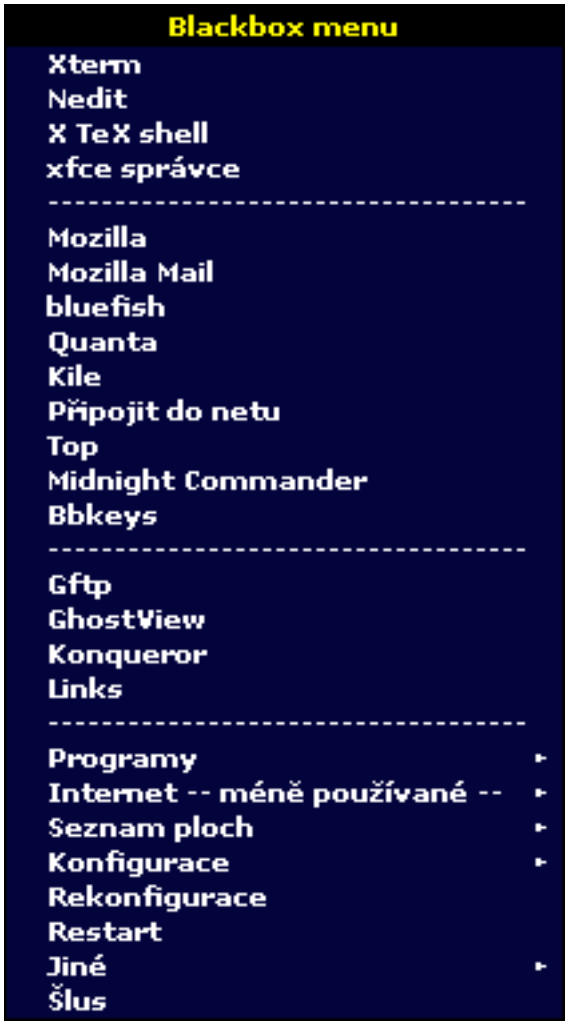


Všechna písma dostupná na Serveru i Stanici.



Všechna písma v kódování ISO8859-2 dostupná na Serveru i Stanici.

Pokud používáte na Stanici jiné prostředí/správce plochy než GNOME nebo KDE (což vřele doporučuji), budou se vám ta písma hodit. Na obrázku je můj Blackbox vyvedený v Microsoft Verdaně (když ještě byla volně přístupná...).



Zvuk

Zásadní slabinou představeného řešení je zvuk. Pokud si totiž pustíte např. xmms, zvuk se ozve (vedle) na Serveru! Jasně, vy **jste** na serveru; nemůže přece hrát Stanice! Ale aby to nebylo tak jasné, chcete-li si na Stanici pustit video (mplayer) ze Serveru, ohlásí vám, že nebylo nalezeno zařízení /dev/video -- protože jste fyzicky na Stanici! Nepochopil jsem, jak to tedy ve skutečnosti je. Která zařízení se použijí ze Serveru a která ze Stanice? Nevím. Všechno, co jsem se zatím dozvěděl, směřuje k faktu, že zatím neexistuje jednotný protokol pro přenos zvuku po síti. Situaci lze řešit za pomoci démonů artsd (z KDE) nebo esd (GNOME), ale nevím jak uspokojivě. Proto jsem vám doporučil, abyste si nainstalovali např. mpeg123: pokud si připojíte adresář s empétrojkami do adresáře na Stanici, můžete hudbu poslouchat lokálně spuštěným programem z konzole (stejně tak např. rádio nainstalované na Stanici uslyšíte jen tam).

Tak a je to!

Pro vaši informaci uvádím údaje o svých strojích a softwaru, abyste si udělali představu:

Server

Původní konfigurace

- K6/2 380 MHz, 96 (128, později 256) MB RAM, 13 GB HDD 5400 otáček, síťová karta 3Com Vortex
- Slackware 8.0, NFS verze 2.0, XFree verze 4.1.0, GNOME 1.4, KDE 2.2.1, Blackbox 0.65 (?)

Nynější konfigurace

- Athlon XP 1700+, 512 MB RAM, 60 GB HDD 7200 otáček, síťová karta VIA Rhine II (integrovaná)
- Slackware 9.0, NFS verze 3.0, XFree verze 4.3.0, GNOME 2.2.0, KDE 3.1, Blackbox 0.65

Stanice

- Compaq Pentium 200 MHz, 32 (později 64) MB RAM (bez swap-oddílu!), 210 MB HDD, FDD, síťová karta 3Com Vortex
- Slackware 8.0; později 9.0, XFree verze 4.3.0

Doufám, že jsem vám svým (víceméně laickým) návodem pomohl k postavení domácí sítě a že vám to nedalo moc námahy. Pokud máte nějaké dotazy, které budu schopen zodpovědět, ptejte se v diskuzi. Ať vám slouží!

FreeBSD v malej firme - 6 (terminálové služby)

Terminály sa vracajú späť - alebo vývoj ide v špirále

Myšlienka terminálu je stará takmer ako počítače samy. V princípe sa jedná o to, ako pristupovať k využívaniu služieb centrálneho počítača pomocou klientskeho zariadenia, pričom klientské zariadenie využíva užívateľ iba na vstup a výstup a samotný program beží na hlavnom počítači. Tieto riešenia (ešte na báze znakových - alfanumerických terminálov) boli s príchodom prvých sálových počítačov bežne využívané a samozrejmé, kým ich nevytlačila éra PC s filozofiou samostatného plnohodnotného počítača na každom stole. Na začiatku sa zdalo, že toto riešenie má spústu výhod. Áno, má, ale iba pre niektoré prípady nasadenia. Žiaľ, pre firemný informačný systém toto usporiadanie prináša veľa nevýhod, či už z hľadiska prevádzky, funkcionality alebo nákladov na prevádzku a udržovanie takéhoto systému v konzistentnom a stále aktuálnom stave. Proste pri nárokoch na firemný IT systém je jednoznačne výhodnejšie, úspornejšie a elegantnejšie riešenie na princípe centralizovaného systému.

Dnes sa teda oprášená myšlienka terminálu po rokoch vracia späť, avšak obohatená o možnosti použitia grafického režimu, čo je pre dnešné firemné riešenia už nutnosť. Výhody tohoto usporiadania sú zjavné:

1. spoločný dátový priestor s hromadným zálohovaním
2. jednoduchá administrácia
3. bezproblémová integrácia celého firemného prostredia do jedného kompaktného celku
4. všetky IT prostriedky je možné bez problémov pružne zdieľať v prípade potreby ktorýmkoľvek z užívateľov a sú stále k dispozícii
5. každý užívateľ sa môže prihlásiť z ľubovoľného stroja a okamžite má k dispozícii svoje aplikácie, svoje dáta aj svoje nastavenie prostredia, samozrejme aj rovnaký výpočtový výkon
6. optimalizácia utilizácie systémových prostriedkov
7. jednoduchšie riešenie havarijných situácií, prestojov a výpadkov
8. ekonomická výhoda takéhoto riešenia - možnosť využitia morálne zastaralej výpočtovej techniky na plnohodnotnú prácu
9. bezpečnosť
10. škálovateľnosť

Aj firmy, ktoré predtým odmietali terminálové riešenia, sa k nim teraz potichu vracajú. Napríklad taký Microsoft a jeho Terminal Server...

Prečo teda terminálové riešenie v malej firme?

Ak sú dáta uložené na jednom mieste, je záloha veľmi jednoduchá, stačí zálohovať dáta na serveri. Pracovná stanica nemá nič, čo by stálo za námahu pri zálohovaní a čo by bolo potrebné nejak extra chrániť (body 1, 9). V prípade potreby sa dá nahradiť ktoroukoľvek inou (viď bod 7).

Výhodné je aj to, že súbory sú na jednom mieste a teda sú v rámci celej firmy informačne konzistentné. Iste každý pozná situáciu, keď je jeden dokument v rôznom štádiu rozpracovanosti na desiatich či viacerých strojoch, užívateľ začne písať do jednej kópie a zabudne na to, že včera písal do inej, čo sa síce volala rovnako, ale bola na inom stroji / v inom adresári (pre usera je to iba drobný nepodstatný detail...). Nakoniec takýto prípad skončí chaosom ("včera niečo napíšem, dnes to tam nemám, zase to vyzerá inakšie") a nastupuje informatik, ktorý sa zúfalo snaži dať dáta dohromady a urobiť z *n* verzií jednu aktuálnu a úplnú... pri jedinom dátovom úložisku a rozumne nastavených prístupových právach sa to pochopiteľne stať nemôže.

Správa takéhoto centralizovaného systému sa potom redukuje na správu centrálneho počítača. Všetko je pohromade a pod kontrolou správcu, useri, aplikácie, dáta, služby. Kdekoľvek sa user prihlási, všade mu systém ponúkne rovnaké, jeho do detailu nastavené prostredie. Každý má presne definované práva. Všetky zdieľané prostriedky sú na serveri, teda ak sa užívateľ úspešne prihlási a má na ne práva, môže ich využívať bez problémov a riešenia zdieľaní, komunikácie a autentifikácie medzi *n* stanicami. Takisto správca môže väčšinou odstrániť problém bez zbytočného behania od stanice k stanici a obojstranne nepríjemného odstavovania userov od práce. Nikdy sa nijaké nastavenie nerozhodí "samo od seba", ani nikto z userov nemá možnosť zasiahnuť do niečoho, čo môže ohroziť systém alebo dáta v ňom. Dáta nikdy neopustia server a neskopírujú sa na klientskú stanicu, klientska stanica prijíma iba výstup z ich spracovania a posiela serveru vstupné údaje, čo je veľké plus pre bezpečnosť citlivých údajov.

Kto sa niekedy staral len o niekoľko Win staníc, chápe rozdiel.

Ako klientské stroje možno použiť staré PC s pamäťou minimálne 64 MB (viac RAMky je samozrejme lepšie, terminály potom neswapujú a systémy im bežia priamo z RAM, naše majú 128 a jeden dokonca 256), procesorom min. Pentium 200 MHz a malým diskom, na ktorom je inštalácia minimálneho základného systému a X. Klientské stroje sú prakticky bezúdržbové a nie je okrem občasnej kontroly konzistencie disku ani nutné ich akokoľvek spravovať. V prípade nasadenia bezdiskových staníc odpadá aj tento posledný krok (tam je zasa nevýhodou o hodne pomalší štart, daný tým, že si musí stanica počas bootu celý systém stiahnuť zo siete). Kto chce a má trochu viac peňazí, môže bootovať aj z Compact Flash ROM a vyhnúť sa tak hrkotajúcemu a potenciálne poruchovému disku, predávajú sa CF adaptéry na bežný IDE kábel.

Za pozornosť stoja body 6, 8 a 10. Veľmi spolu súvisia. Ak systém prestane zvládať požadovanú záťaž, stačí vymeniť server za výkonnejší a ide sa ďalej naplno, na rýchlosť systému nemá klientská stanica takmer nijaký vplyv. Výkon skôr ovplyvňuje rýchlosť prenosu dát po sieti, latencia prenosu (teda časové oneskorenie - doba od vyslania dát po ich doručenie) a samozrejme výkon servera samotného. Preto sa po určitej dobe nemusia vyraďovať staršie PC, hoci by už nespĺňali požadované HW parametre na samostatnú prevádzku, na funkciu zobrazovacej vstupno-výstupnej jednotky majú stále výkonu dostatok a možno ich použiť prakticky dovtedy, dokedy slúžia. To je veľmi výhodné aj z hľadiska nízkych nákladov na zostavenie a upgrade takéhoto riešenia. Navyše server plynule rozdeľuje výpočtový výkon tým procesom, ktoré ho potrebujú, a preto aj keď je vyťažený takmer naplno, užívatelia pracujú plynule a rýchlo. K tomu však netreba kupovať *n* výkonných strojov (ktoré by po 80 percent času iba stáli a počítali šetrič a iba 20 percent ich výkonu by sa skutočne využilo na prácu), ale iba jeden. Teda výhoda investičná aj prevádzková zároveň...

Licencie komerčných riešení sa môžu predražiť

Silným dôvodom pre nasadenie unixového terminálového systému je aj cenová a licenčná politika MS - je rozdiel kúpiť trebárs 5 samostatných plne vybavených PC pracovných staníc, ku každej z nich OS Windows + kancelársky a iný software, a kúpiť 5 lacných starších počítačov iba za cenu HW, bez OS, bez platenia softwarových licencií na každú inštaláciu. Takto možno aj s malým rozpočtom postaviť veľmi solídny firemný systém. Celá vec stojí (a padá) len na fantázii a schopnostiach admina - tvorcu firemného informačného systému.

V prípade Microsoft Terminal server to nie je žiadna výhra - majiteľ systému musí zaplatiť cenu licencie OS Windows v klientskom stroji (z niečoho ten TS klient predsa spustiť musí), licenciu MS Server 2000, licenciu MS Terminal Server a potom ešte licenciu z každého sedenia (per seat). A má holý systém bez aplikácií, ktorého klientske stanice vyžadujú úplne rovnakú starostlivosť ako samostatné stroje (ten Windows treba pravidelne čistiť od temporaries, defragmentovať, udržiavať aktuálny antivírus atď). Jediným zachovaným plus z celej terminálovej koncepcie tu ostáva to, že stanice používajú výpočtový výkon servera.

Ekonomický aj prevádzkový efekt nasadenia terminálov v prostredí MS sa teda viacmenej stráca a pre malú firmu je to dokonca silne nevýhodné riešenie... záujemcom doporučujem ako povinné čítanie cenníky Microsoftu.

Obdobná situácia je, keď sa miesto MS Terminal Servera kúpi Citrix - všetky hore uvedené veci (až na iný typ terminálovej nadstavby MS servera) platia opäť.

Dajme užívateľom to najlepšie!

Ešte jedna poznámka, skôr psychologická ako odborná - aby užívateľ nemal pocit, že sedí len pred nejakým "orezávatkom" a že mu admin hodil na stôl ten posledný šrot z celej firmy, je vhodné dať terminálom pokiaľ možno veľké monitory a optické myši (a samozrejme aj skrinky PC vyčistiť od prachu, nálepiek a inej špiny, aby vyzerali k svetu). Z hľadiska prevádzky je to síce bez významu, ale z hľadiska akceptácie nového systému nedôverčivým Win užívateľom to má význam obrovský. Ťažko niekoho nadchne stará zaprášená kraksňa s blikajúcim 15" monitorom. Zato keď niekomu postavím na stôl monitor devätnástku s perfektným ostrým obrazom, na ktorom je radosť čítať, to je niečo iné... A keď ešte spozná tú rýchlosť práce, nazad ku Windowsu sa mu nebude chcieť. Monitor 19" sa dá v bazári zohnať za pár tisíc. Celá zostava sa v prípade bazárového predaja dá kúpiť tak za 4 až 5 tisícok Sk. Porovnajme si to s cenou novej PC zostavy...

Topológia siete

Pre terminálovú sieť je optimálne a najčastejšie využívané hviezdicové usporiadanie, kde základ siete tvorí switch, na ktorý sa pripájajú stanice aj server. Nedoporučujem používať hub, pre rýchlu prácu s xdmcp reláciami je potrebné mať malú latenciu a pokiaľ možno vysokú priepustnosť siete. Tu má switch oproti hubu výhodu, že posiela pakety iba príjemcovi a ostatní účastníci siete nie sú bombardovaní hromadou cudzích paketov, ktoré by ich spomaľovali a zbytočne zvyšovali traffic. Navyše je to aj výhoda z hľadiska bezpečnosti. V praxi sa osvedčil malý osemportový 100 MBit switch, chodí to rýchlo a odozvy terminálov mám v reálnom čase. Plánujem rozšírenie na 16 portový rackový switch.

Udržanie nízkej latencie komunikácie je pre X dokonca dôležitejšie ako prenosová rýchlosť samotná. Skúšal som pripojiť terminál cez 11 MBit WiFi a napriek tomu, že šírka pásma stačila a nebola využitá "na doraz", oneskorenie prenosu bolo tak strašné, že sa dalo robiť iba s veľkým sebazaprením... niekedy človek videl reakciu na úkon, ktorý urobil, až po pár sekundách. Použiteľné to bolo snáď iba na núdzové účely, rozhodne nie na serióznu prácu.

Konfigurácia na strane servera (X klienta)

Pre rozbehnutie terminálových služieb X je potrebné nastaviť na serveri (tu sa myslí počítač, na ktorom aplikácie fyzicky pobežia, z hľadiska terminológie systému X window je to X klient) konfiguračný súbor správcu prihlásenia, aby umožňoval pripojovanie vzdialených sedení cez protokol xdmcp (X Display Manager Communication Protocol). V našom prípade sa použil správca prihlásení kdm. Jeho konfiguračný súbor sa nazýva `kdmrc` a nájdeme ho v adresári `/usr/local/share/config/kdm`. Súbor má veľmi ľahko pochopiteľné názvy premenných a jeho jednotlivé voľby teda nie je problém nájsť a nastaviť. Pre povolenie vzdialených sedení je treba nastaviť sekciu `[Xdmcp]` takto:

```
[Xdmcp]
Enable=true
Port=177
Willing=/usr/local/share/config/kdm/Xwilling
Xaccess=/usr/local/share/config/kdm/Xaccess
```

Prvý riadok zapína xdmcp protokol, v druhom sa určuje, na akom tcp porte má X komunikovať. Pre xdmcp v firemnej LAN sieti sa doporučuje nechať tam výchozí port 177. Meniť to má význam iba v odôvodnených prípadoch, napríklad keď chceme riešiť bezpečné pripojenie vo verejnej sieti tunelovaním protokolu xdmcp cez ssh. Tretí riadok hovorí o ceste na skript `Xwilling`, ktorý umožňuje samotné vzdialené prihlásenie. Pre bezpečnosť je podstatný štvrtý riadok, ktorý ukazuje cestu na súbor `Xaccess`. Jedná sa o konfiguračný súbor v tom istom adresári ako `kdmrc`, ktorý povoľuje prístup na X jednotlivým IP adresám. Po inštalácii je nastavený tak, aby povolil prihlásenie akéhokoľvek stroja, ktorý o vytvorenie vzdialenej X session požiada. Jeho editáciou môžeme zakazovať a povoľovať prístupy jednotlivým klientským staniciam.

Ešte tri poznámky. Z bezpečnostných dôvodov je zakázaný login roota do KDE. Ak ho z nejakých príčin potrebujeme , dá sa to urobiť v `kdmrc` v sekcii `[X-*-Core]` cez voľbu `AllowRootLogin=true` - pôvodne je tam nastavené `false`.

Druhá dôležitá vec - je dobré na serveri zakázať možnosť vypnúť počítač zo vzdialenej X session. Predídeme tým nepríjemnostiam, keď nejaký "snaživec" namiesto ukončenia sedenia vypne server a s ním aj ostatných užívateľov s rozrobenou prácou. Toto je možné najjednoduchšie urobiť v ovládacom centre KDE, kde v sekcii Správca prihlásenia nastavíme, že shutdown je povolený iba z lokálneho prihlásenia. Samozrejme je možné nastaviť to aj editáciou `kdmrc`, kde povolíme shutdown z lokálu a zakážeme ho z vzdialeného prihlásenia.

Tretia poznámka, ktorú som už uviedol v časti Inštalácia, pre každý prípad ju však opakujem. Z mne neznámeho dôvodu protokol xdmcp vo FreeBSD nechodí bez povolenia IPv6 v konfigurácii siete na strane klienta aj servera. Sám som strávil dva týždne rozmýšľaním nad tým, prečo mi terminál nepripojí vzdialený server. Potom som narazil na popis chyby na jednej konferencii o FreeBSD a KDE. Riešenie popisované v konferencii bolo jednoduché, povoliť IPv6. Po nastavení `ipv6_enable="YES"` v `/etc/rc.conf` na serveri aj termináli sa terminál okamžite pripojil. Zrejme blokovanie IPV6 nechcene zablokuje aj xdmcp... ale môže to byť len môj dohad. Každopádne to po príslušnom nastavení funguje a chodí bez chyby.

Konfigurácia na strane terminálu (X servera)

Na strane X terminálu (v terminológii X window sa nazýva X server) je iba potrebné spustiť X s parametrom `query` na daný server. Keďže od obyčajných userov nemôžem chcieť, aby sa prihlásili lokálne do konzoly a odtiaľ si ručne spustili vzdialenú reláciu X, musíme tento proces automatizovať. Mne sa osvedčilo jednoduché riešenie. Do adresára `/etc/rc.d` treba vytvoriť súbor s názvom `X.sh`. Je to jednoduchý shellový skript, ktorý obsahuje iba dva príkazy:

```
/usr/X11R6/bin/ X -query 10.10.10.10
shutdown -p now
```

kde 10.10.10.10 je adresa nášho terminálového servera. Po štarte systému na terminálovej stanici automaticky nabehne vzdialené prihlásenie na server a objaví sa nám rovnaké prihlasovacie okno ako na serveri samotnom. Ak chce užívateľ skončiť prácu s terminálom, ukončí beh X na termináli cez Ctrl+Alt+Backspace a ihneď sa spustí ukončovacia sekvencia, ktorá zastaví systém a (v prípade, že to matičná doska umožňuje a táto vlastnosť je u nej podporovaná) automaticky vypne napájanie terminálu. Ak nie, iba ukončí systém a treba terminál ručne vypnúť.

To by bolo snáď všetko. Ak ste nastavili všetko tak, ako treba, mal by vám bez problémov bežať terminálový systém. Zatiaľ sa lúčim s pozdravom: nech vám vaše systémy idú!

Související články

FreeBSD v malej firme - 1 (Ľahko a bez námahy)
FreeBSD v malej firme - 2 (Inštalácia)
FreeBSD v malej firme - 3 (RAID a Xserver)
FreeBSD v malej firme - 4 (KDE, lokalizácia)
FreeBSD v malej firme - 5 (tlačové služby)

Práva

Každý uživatel má dána práva pro přístup, čtení a zápis souborů či složek. Vlastníkem souboru/složky je ten kdo soubor/složku vytvořil. Vlastníkem je i skupina ve které je daný uživatel, ta však nemůže měnit jeho majitele. Práva k souboru/složce zistíte příkazem
ls -l
, kde prvních 10 znaků udává přístupová práva:
Pokud první znak obsahuje písmeno "d" jedná se o složku (adresář-dir). Dalších 9 zanků se dělí na 3 skupiny po třech. Kde:

- 1. skupina udává práva uživatele
- 2.skupina práva skupiny ve které uživatel je
- 3.práva všech ostatních

Každá skupina (1., 2. i 3.) obsahuje 3 znaky ty znamenají:

- **r** - (read) - práva pro čtení souboru / výpis souborů či podadresáři ve složce
- **w** - (write) - práva pro zápis do souboru / přidávat či mazat soubory ve složce
- **x** - (execute) - práva pro spouštění souborů / zpřístupňovat soubory či podadresáři ve složce
- **-** - bez práv (chybí pouze to právo kde by mělo být r,w nebo x)

Pokud Vám chybí všechna práva r,w,x nemůžete ani vstoupit do složky (cd). Pokud kopírujete soubory z jednoho místa na druhé, práva se zachovávají. Příklad:
drwxr-x--x 4 lolek student 4096 Jun 16 16:41 test
Pořadí písmen:

- 1. **d** : Jedná se o složku (s názvem test, patřící uživateli lolek a skupině student)
- 2. **r** : uživatel lokel smí číst soubory ve složce umístěné
- 3. **w** : uživatel lokel smí zapisovat a mazat soubory
- 4. **x** : uživatel lokel smí otevřít složku (pokud mu chybí právo r-nesmí však číst soubory v ní umístěné, pokud w - nesmí je vytvářet ani mazat)
- 5. **r** : skupina student smí číst soubory ve složce umístěné
- 6. **-** : skupina student nesmí vytvářet ani mazat soubory
- 7. **x** : skupina student smí otevřít složku
- 8. **-** : ostatní mimo skupinu nesmí číst seznam souborů ve složce
- 9. **-** : ostatní mimo skupinu nesmí zapisovat a mazat soubory
- 10. **x** : ostatní mimo skupinu smí otevřít složku

Změnut práva můžete příkazem **chmod**, vlastníka souboru **chown**, skupinu vlastníků **chgrp**.
Vlastník a skupina se dá změnit najednou pomocí `chown`:
chown uživatel:skupina soubor adresář/ a rekurzivně s přepínačem `-R`

Tento dokument vznikl přepisem **Učebnice GNU/Linuxu** z www.abclinux.cz
Dokument je uvolněn pod licencí GNU/GPL.