# 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. 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

# Stručný výtah

Linux, či přesněji GNU/Linux je moderní operační systém. Skládá se z jadra 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 procesor 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 připravovaná verze 2.6 jej posune 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.

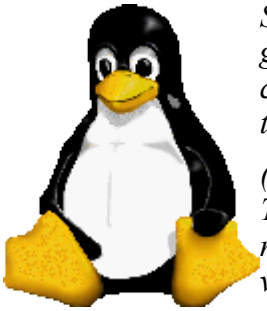
# Podpora

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 repozitář 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

# 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 předávájí 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` můžezete 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 získaly 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 používate vstupní a výstupní zařízení (klávesnci, 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.

## Zadávani 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 zdali 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 spusti 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 věžte, 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 soubour 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 upředňostňují bash, zatímco *BSD spíše csh, nebo jeho rozšířenou a zpětně kompatibilní "nádstavbu" - 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:~$` - takto, nebo velmi podobě Vás bash přivítá. Místo `uzivatel` bude vase uzivatelské jméno a localhost bude název počítače. (V případě, že před tím vyplňujete username a password, tak to ještě nejste v shellu.) Práce s bashem je velice jednoduchá - zadáte příkaz a odentrujete. Pro orientaci v příkazech poslouží Přehled příkazů a sekce Principy práce se systémem. Pokud změníte složku mimo svou domovskou, bash vypise `uzivatel@localhost:/slozka$` 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 osubory 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 témnatu, 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 stejnojmeným programem. Info stránky jsou rozsáhlejší, strukturované (info stránky podporují hypertextové odkazy). Spousta manuálových stránek obsahuje upoornění, 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 rourou `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 (pkud 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 '**x**' značí, že je možné vypsat seznam souborů. 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 vyvtořil (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
```

### 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 vyjímkou 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 mohli 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.

## 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 (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
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 možná i nějaká další distribuce). 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 zmáčnout `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 nepoužitené, 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.

Provotní 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 visuálních 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 ukonči 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 superužitele.

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 používate 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 používate 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 rozodně 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 vývojářu 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 domívají, že uživatelé jsou málem negramotní.

### XFCE


Pokud chcete hezký a funkční desktop při nízké spotřebě systémových zdrojů, tak XFCE je presně 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 vlatní 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 (těmi se myslí Pentia okolo 150 MHz a 32 MB RAM). Mezi hlavní přednosti Fluxboxu patří až neuvěřiteně 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 nazvájem 100% komatabilní. 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 programky 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"

Všecny 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 nezjnámnější patří GTK a QT.

### Qt

Aplikace napsané v Qt zapadnou do výše popisovaného prostředí KDE, které v něm je taktéž napsáno. Jako programocí 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 programatorský 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 obyvykle využívají 2D či 3D knihoven pro práci s grafickými funkcemi a akcelerací grafické karty.

Nejznámnějšími 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 tučnáka 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 kopíruje autentizační cookie z  provede automaticky. 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 plikací 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 kvalitni 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 příliž 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 "fukncí" 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 samosebou 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 filteru. 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 retětez - `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 bytl 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 příliž velké pakety některá zarízení neumí přenést, lze tak učinit pomocí přepínače `--fragment` (zkráceně `-f`).

Poté, co jsme probrali nutné minimum k `iptables`, obrátíme pozoronost 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 smurf-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

Využité místo na disku.

## 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,...

# Dodatky

## 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.

Puvodne platila rovnice kernel = Linux. Tedy Linus Torvalds napsal jadro operacniho systemu, ktere konzistentne **obaluje hardware** a poskytuje zakladni stavebni kameny pro aplikace (**userland**), ale samo o sobe neprinasi uzivateli zadny primy uzitek.
Po spojeni s projektem **GNU** (GNU's not UNIX) Richarda Stallmana se zacalo pod pojmem Linux oznacovat **cele prostredi** vcetne aplikaci (tedy Linux = kernel + GNU). Vydavani Linuxu se pote chopili ruzni **distributori**, kteri do kernelu i programu z projektu GNU zacali pridavat sva (misty komercni/uzavrena) **rozsireni**. Situaci vystihuje oznacovani jmena distribuce Debian GNU/Linux (tedy Debian = kernel + GNU + pripadne vlastni upravy distributora).

Existuje tzv. **vanilla kernel** ("od Linuse"), ktery mel az do verze 2.6.8 cislovani ve tvaru MAJOR.MINOR.SUB (licha MINOR jsou vyvojove verze), pozdeji ve tvaru MAJOR.MINOR.SUB.FIX (vsechna MINOR jsou stable, FIX jsou bugfixy posledniho SUB) a **odvozene verze** (defacto patchsety) vyznamnych vyvojaru, ktere jsou od vanilly odvozene. Poznaji se podle toho, ze konci pomlckou a dvema pismeny: -ac (Alan Cox), -mm (Andrew Morton) apod. Tyto upravene verze pak obsahuji zmeny, ktere se jeste nedostaly do **hlavniho stromu jadra**, ale mohou byt uzitecne pro spoustu uzivatelu. 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 chběla 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

**R**ichard **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 - II (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

**G**NU **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, vyhledavani 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**ile**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.

# Souborové systémy - často kladené otázky

# 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.

Michal Vyskočil, 18.6.2004 13:15
Upravit

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.

**Přepracoval:**

Pavel Vlasák

www.aboutme.ic.cz