Tvorba distribuce OS Linux

Building Linux distribution

Martin Zajíc

Bakalářská práce
2011

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ABSTRAKT

Tato bakalářská práce v teoretické části definuje základní pojmy, jako jsou Linux, dále pak linuxové distribuce, jazyk BASH, zabezpečení operačního systému Linux a LiveCD. Úvodní část je věnována informacím o systému Linux a distribuci Linux From Scratch. Dále pak obsahuje stručný úvod do jazyka BASH a kompilaci zdrojových kódů včetně možností sestavení Linuxových distribucí. Na závěr teoretické části je popsáno zabezpečení Linuxu, hodnocení bezpečnosti Software a tvorba LiveCD.

V praktické části je rozebrána struktura scriptů a funkce vytvořené v jazyce BASH. Dále je uveden návod na sestavení distribuce LFS pomocí vytvořených skriptů. A na závěr uvedeny zdroje pomoci při chybách v kompilaci a možnosti zlepšení skriptů.

Klíčová slova: Linux, distribuce, BASH, LFS, LiveCD, bezpečnost Linuxu, kompilace, DAC, MAC

ABSTRACT

This bachelor work includes the terms such as Linux in it's theoretical part. Further, the terms such as Linux distribution, BASH language, operating system security, and LiveCD. The introductory part is dedicated to the information about the Linux system and Linux From Scratch distribution. Then it also includes brief introduction to the BASH language and the source code compilation, including the possibility to compile Linux distributions. At the end of the theoretical part, there is described Linux security, the evaluation of Software security and the LiveCD creation.

In the practical part, there is analysed the script structure and the functions written in BASH language. Next, there is the manual for compiling LFS distributions using the generated scripts. And at the end of this part, there are mentioned resources to compilation errors and the possibilities of script improvement.

Keywords: Linux, distribution, BASH, LFS, LiveCD, Linux security, compilation, DAC, MAC

Chtěl bych poděkovat především vedoucímu mé bakalářské práce, panu doc. Ing. Martinu Syslovy, Ph.D., za jeho cenné rady a připomínky. Dále bych chtěl poděkovat celé mojí rodině za její obrovskou podporu během celého mého studia a také mým přátelům, se kterými jsem prožil tři roky studia na Univerzitě Tomáše Bati.

”Nejlepší knihy jsou takové, které člověku říkají, co už sám ví.”

GEORGE ORWELL

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo –bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně .....

podpis diplomanta

OBSAH

ÚVOD

Operační systém Linux je v dnešní době k nalezení ve všech možných druzích elektronických zařízení od náramkových hodinek až po automobily. Pro administrátora Linuxového operačního systému je důležité znát tento systém mnohem podrobněji než např. konkurenční Microsoft Windows, protože je vhodné si operační sytém Linux nastavit do nejmenšího detailu. Jedním z výborných dokumentů, díky kterému se dá zjistit o Linuxu velké množství informací, je kniha Linux From Scratch, která popisuje sestavení Linuxového operačního systému. V knize jsou uvedeny informace, postupy sestavení a hlavně účel jednotlivých programů a balíčků programů, díky čemuž získá uživatel větší představu o tom, co se v systému nachází a k čemu je to dobré. Automatické skripty pro sestavení jsou pak spíše určeny uživatelům, kteří již tyto informace znají a chtějí sestavit základní systém bez zbytečného zdržování.

Bakalářská práce je rozdělena do dvou částí - teoretické a praktické. V teoretické části jsou definovány základní pojmy, jako Linux, dále pak distribuce operačního systému Linux, liveCD, jazyk BASH, kompilace zdrojových kódů, způsob vytváření distribucí a zabezpečení Linuxového operačního systému. V praktické části je pak rozebrána struktura skriptů pro tvorbu distribuce Linux From Scratch, popsány některé funkce, které jsou ve skriptech použity, popsán návod pro sestavení pomocí těchto skriptů a také nastíněny možnosti vylepšení.

I. TEORETICKÁ ČÁST

1 LINUX A SVĚT KOLEM NĚJ

1.1 OS Linux

Linux je v informatice označení pro unixový operační systém (původně pouze jeho jádro). Linux je šířen v podobě distribucí, které je snadné nainstalovat nebo přímo používat (tzv. LiveCD). Zároveň se díky použitým licencím jedná o volně šiřitelný software, takže je možné ho nejen volně používat, ale i dále upravovat a distribuovat (kopírovat, sdílet). Tím se odlišuje od proprietárních systémů (např. Microsoft Windows či Mac OS X), za které je nutné platit a dodržovat omezující licence. [6]

Obr. 1. Typický obrázek tučňáka spojovaný s Linuxem

1.2 Dnešní použití OS Linux

V současnosti je Linux součástí trhu se stolními počítači. Zpočátku se vývojáři Linuxu zaměřovali na síťové systémy a služby, kancelářské aplikace představovaly poslední bariéru, kterou bylo nutno překonat. Na tomto trhu dominuje Microsoft, proto v posledních několika letech vznikala spousta alternativních projektů, jejichž cílem je nabídnout Linux jako vhodnou volbu na pracovní stanice. V rámci těchto projektů vznikají snadno použitelná uživatelská rozhraní i kancelářské aplikace (textové editory, tabulkové procesory,…), kompatibilní s aplikacemi Microsoft Office (LibreOffice, Koffice, AbiWord,…) [5].
V oblasti serverů, má Linux pověst stabilní a spolehlivé platformy, na které běží databázové a další služby takových společností jako je Amazon, americká pošta [17], německá armáda [16], Google, Facebook a další. Velmi oblíbený je Linux u poskytovatelů internetového přístupu a služeb, kde se používá jako firewall, proxy server nebo webový server. Počítač s Linuxem najdete i u každého správce některého Unixového systému, který jej používá jako pohodlnou administrativní stanici. Clustery linuxových počítačů se podílely na vzniku filmů jako Titanic [46] nebo Shrek [48] [47]. Na poštách slouží jako centrály řídící směrování zásilek, velké vyhledávací stroje pomocí nich prohledávají Internet. To je jen ukázka několika z mnoha tisíc náročných úkolů, které dnes Linux na celém světě vykonává [5].
Stojí také za zmínku, že moderní Linux běží nejen na pracovních stanicích, středních a velkých serverech, ale i na ”hračkách”, jako jsou PDA či mobilní telefony, ve spoustě zařízení spotřební elektroniky, a dokonce i v experimentálních náramkových hodinkách [49]. Linux je jediný OS na světě, který pokrývá takto širokou škálu HW. [5]

1.3 Co je to distribuce

Distribuce, pokud je řeč o distribucích operačního systému, je balíkem jádra operačního systému a doplňujícího SW. Distribuce nejsou jenom záležitostí Linuxu, i když jsou často mylně spojovány pouze s Linuxem. Kromě Linuxových distribucí existují například distribuce s jádrem BSD (OpenBSD, NetBSD, FreeBSD,…), HURD (ArchHurd), XNU (PureDarwin) a mnoho jiných. Přídavný SW distribuce je čistě volbou autora distribuce, může se tak jednat například o SW projektu GNU (nejvýznamnějšími komponenty jsou GCC, glibc, Emacs nebo GNOME) [18], SW projektu BSD, proprietární (uzavřený) SW nebo jakýkoliv jiný SW jehož licence dovoluje použití v distribuci.

1.4 GNU a Linux

Rekurzivní akronym pro GNU is Not Unix, kde GNU znamená GNU is Not Unix,… Projekt založený v roce 1984 Richardem Stallmanem a společností Free Software Foundation na vytvoření svobodného a otevřeného operačního systému (dále jen OS) na základě OS UNIX. Zpočátku soubor systémových programů, kterým chyběla hlavní součást - jádro (kernel). Později se jako jádro použil projekt Linuse Torvaldse Linux. V mínění veřejnosti se společný OS GNU/Linux přejmenoval pouze na Linux podle jádra. Jedna z mála distribucí, které dodržují správné pojmenování tohoto OS, je Debian GNU/Linux. GNU obsahuje příkazy každodenní potřeby jako tar, awk, top, dd, ale třeba i Samba, xfig apod. De facto veškerý software vydávaný v rámci projektu GNU je licencován GNU General public license (GPL). Dodnes je vyvíjeno původní microkernel jádro GNU/HURD, avšak ani za 21 let vývoje jej vývojáři nedokázali dovést ke stabilnímu vydání. [19] [18] [20]

2 LINUXOVÉ DISTRIBUCE

Linuxová distribuce je v informatice označení pro snadno použitelný Linuxový systém, přičemž název je odvozen od jádra Linuxu, které je základní součástí každé distribuce. Distribuce jsou vytvářeny proto, aby uživatel nemusel jádro a doplňující software sám náročným způsobem skládat do funkčního celku. Obsažený software je volně dostupný na Internetu (typicky open source software). Pro odlišení jsou distribuce pojmenovány (např. Ubuntu, Fedora, …), přičemž každá je jinak zaměřena (pro nezkušeného uživatele, pro vývojáře, výuku atp.).[10]

2.1 Hlavní odlišnosti

Linuxové distribuce se liší v mnoha více či méně zásadních ohledech. Nejvýraznějšími rozdíly, které dělají distribuce distribucemi jsou: cyklus vydání, balíčkovací systém, základní SW výbava, nasazení,…

2.1.1 Cyklus vydání

V zásadě rozlišujeme 3 vydávací cykly:

Dlouhý cyklus vydání Tento cyklus vydání je delší než jeden rok. Používají ho distribuce, které nezakládají na aktuálnosti SW, ale na jeho stabilitě. Především pak **Debian (stable)** a **Red Hat Enterprise Linux** (dále pouze RHEL). Vývojáři Debianu před nedávnem (7. 2009) rozhodli o tom, že bude Debian vydáván v pravidelných intervalech a to tak, že na konci každého lichého roku dojde k zmrazení větve *testing*, ta je následně nějaký čas testována, aby byly odstraněny všechny bugy, po jejichž odstranění je prohlášena za stabilní a balíčky v ní obsažené jsou přesunuty do větve *stable* [1)]. RHEL vychází v nepravidelných cyklech (cca 2-3 roky) a v mezičase vychází updaty. [2)]

Krátký cyklus vydání Typicky půl roku až rok. Používá ho většina deskopových distribucí. Tento vydávací cyklus je kompromisem mezi stabilitou a aktuálností

[1)] časový odstup mezi vydáním verze 5 lenny a aktuální 6 squeeze byl 2 roky. Nejdelší odstup mezi vydáními byl 3 roky a to mezi vydáním 3.0 woody a 3.1 sarge

[2)] časový odstup mezi verzí 5 a 6, byl 3.5roku a v mezičase vyšlo 5 update verzí.

SW. Např. **Ubuntu** [3], **Mandriva** [4](dříve Mandrake), **Fedora** [5],…

Průběžné aktualizace (rolling-updates) Distribuce používající rolling-updates se vyznačují především aktuálností SW, většinou udržují větev pro uživatele a testovací větev, kde je SW podroben krátkému testovaní, než se přesune do uživatelské větve. Tyto distribuce většinou vydávají jednou za čas instalační liveCD aby bylo možné je nainstalovat i na aktuální HW. **Gentoo**, **Arch Linux**, **Debian sid (experimental)** [6],…

2.1.2 Balíčkovací systémy

Balíčkovací systémy jsou standartní součástí nejen Linuxových distribucí. Balíčkovacím systémem je i systém aktualizací MS Windows nebo oblíbené úložiště aplikací (markety) ve smartphonech (Android market, BlackBerry App World, Ovi Store, Windows Marketplace,…).

DPKG *Debian package management system* Jedná se o základní „nízko úrovňový" balíčkovací systém pro debian, umožňuje pouze instalaci mazání a výpis balíčků (balíčky jsou vždy s příponou deb).

Nejčastěji se používají programy, které mají rozšířenou funkcionalitu, jsou založené a volají dpkg. Nejznámějším je **APT** *(Advanced Packaging Tool)*[11], od apt se upouští a doporučuje se používat jeho frontend **Aptitude**, protože aptitude dokáže lépe řešit konflikty balíčků, poskytuje jednotné rozhraní pro správu balíčků[7] a GUI založené na ncurses. Pro tento balíčkovací systém existuje i velké množství GUI: synaptic (GTK), Ubuntu Software Center (GTK),KPackage (Qt),…
Tento balíčkovací systém používájí kromě již zmíněného debianu i jeho deriváty jako je Ubuntu, ale i další distribuce a OS. Existuje port pro MacOSX nebo Opensolaris[11].

RPM *Red Hat Package Manager* Druhý nejčastěji používaný balíčkovací systém. Je používán množstvím velkých distribucí v čele s RHEL (Red Hat Enterprise Linux), také je používán distribucemi Fedora, Mandriva, SUSE, ArkLinux,...

RPM označuje jak název základního balíčkovacího systému tak i název balíčků samotných (vždy s příponou rpm). RPM se především vyznačuje nepřenosností z jedné distribuce na druhou (na rozdíl od deb balíčků, kde to více méně funguje). Proto se lze často setkat s názorem, že rmp neumí pořádně řešit závislosti. Tímto názorem se většinou vyznačují lidé, kteří kombinují repozitáře různých distribucí založených na RPM.

RPM se často vyznačuje ještě jedním specifikem a to tím, že na rozdíl od DPKG, kde se nejčastěji používají obrovské repositáře s obrovským množstvím balíčků se u RPM používá velké množství repositářů s už ne tak velkým množstvím software. Např. Debian používá jeden repozitář pro každou jeho verzi (Stable, Testing, Unstable) když to u RPM distribucí se setkáme s repozitáři zvlášť pro multimédia, core system, apd...

[3] **Ubuntu**: cyklus vydání je půl roku, u jeho derivátů pak zpravidla o něco málo delší. Vydaní probíhá pravidelně v dubnu a říjnu, přičemž každé dva roky vyjde LTS verze s podporou na dva roky. Poslední verze: 11.04

[4] **Mandriva**: cyklus dlouhý půl roku, vychází verze s číslem roku a verze spring. Poslední verze: 2010 Spring

[5] **Fedora**: cyklus vydání každého půl roku. Poslední verze: 14

[6] testovací větev debianu, která se jmenuje Sid, používá právě rolling-updates, po otestování SW obsažený ve větvi putuje do větve *testing*

[7] APT je několik aplikací (apt-cache, apt-get,…), kde každá poskytuje jinou funkci

PACMAN Jedná se pravděpodobně o jeden s nejrychlejších balíčkovacích systémů a je součástí distribuce ArchLinux jejích derivátů larch, FaunOS, Archie, Chakra a je také používán v distribuci DeLi Linux a Frugalware Linux (ve Frugalware linux byl vytvořen fork a jeho vývoj probíhá odděleně). Pacman má výhodu proti jiným balíčkovacím systémům nejen v rychlosti, ale také v tvoření balíčků. Na rozdíl od DPKG, kde existují rozsáhlé návody jak vytvořit balíček, stačí u Pacmana vytvořit podle jasných pravidel skript s názvem PKGBUILD, který obsahuje všechno od licencí, závislostí, zdrojů kontrolních součtů až po návod jak postupovat při sestavení balíčku, poté stačí napsat makepkg a balíček se sám vytvoří bez jakéhokoliv zásahu.
Na tomto principu fungují jak source repoziráře v ArchLinuxu zvané ABS tak i uživatelské repozitáře AUR.

Pro pacman existuje velké množství wrapperů[8)] pacman-color (přidává do výstupu barvy). Defaultní Pacman také neumí pracovat uživatelským repozitáři AUR proto vzniklo několik wrapperů, které přidávají podporu vyhledávání a instalace balíčků z AUR např. Clyde nebo yaourt. Jiné wrappery zase přidávají podporu stahování z více zdrojů najednou aby byl balíček stáhnut co nejdříve např. PowerPill nebo Bauerbill.

Všechny tyto programy pracují s repozitáři a balíčky stejně, jsou tedy kompatibilní a mají dokonce i stejnou syntaxi.

Existují i GUI pro pacman, nejznámější je asi shaman (Qt).

Portage Portage je balíčkovací systém GNU/Linuxové distribuce Gentoo Linux, který je podobný systému portů z FreeBSD. Portage je napsán v programovacím jazyce Python. Hlavním příkazem je zde příkaz emerge. Tento balíčkovací systém využívá balíčky ebuild, které zajistí zkompilovaní podle nastavených systémových proměnných jako jsou například USE, CFLAGS, CHOST a LINGUAS. Portage automaticky dohledá závislosti, stáhne požadované balíčky a program nainstaluje.

Existují různé grafické nadstavby, např. Kuroo (Qt) nebo Porthole (GTK).

Další,... Další balíčkovací systémy je např. **slapt-get** (balíčkovací systém, který používá nejstarší dodnes vyvíjená distribuce Slackware a její deriváty (BackTrack, VectorLinux,Slax,...) balíčkovací systém používá obyčejné Tar balíčky komprimované Gzipem), **Equo**, **Smart Package Manager**,…

2.1.3 Nasazení

Linuxové distribuce se dále liší podle druhu zařízení pro která jsou určena. Nejčastěji je to server nebo desktop, ale existují i jiné druhy zařízení na kterých linux běží.

User choise (volba uživatele) Některé distribuce si nedělají starosti s určením na kterém druhu zařízení budou použity. Poskytují pouze základní sadu nástrojů (programů), které uživatel doplňuje podle svých požadavků. Toto je typická vlastnost Debianu (u debianu už dnes existují přímo různé konfigurace distribuce jako server, desktop nebo laptop), Archlinuxu, LFS nebo Gentoo.
Tyto distribuce je možné většinou nasadit na všechny níže zmíněné zařízení.

Desktop Dalšími zařízeními pro které je linux určen jsou desktopové (personální) počítače. Tyto distribuce poskytují grafické programy pro běžné použití např. v kanceláři. Typicky např. Ubuntu (desktop), Fedora, Mandriva,…

Server Serverové distribuce obsahují základní nástroje pro nasazení na serverových počítačích a neobsahují na rozdíl od desktopových distribucí grafický server Xorg, ale obsahují serverové nástroje jako je např. webový server apache. Typicky serverovou distribucí je např. Ubuntu (server), CentOS, RHEL,…

[8)] wrapper je program, který funguje pouze s jiným programem a rozšiřuje jeho funkcionalitu

Enterprise Enterprise distribuce jsou distribuce určené pro podnikové nasazení a na rozdíl od běžných distribucí mají placenou podporu ze strany dodavatele a delší testovací dobu např. SLED (SUSE Linux Enterprise Desktop) nebo RHEL.

Embedded Jedná se o distribuce směrované na určité zařízení. Nejznámější je např. openwrt, což je distribuce určená pro síťové nasazení, např, routery, gatewaye apd,…
Linux, ale najdeme i v embedded zařízeních jako jsou settopboxy, televize, IP kamery, pračky, mikrovlné trouby, chladničky a jiné jednoúčelové zařízení,…

Mobilní Asi nejnovějším prostředí ve kterém momentálně již systém linux dominuje, jsou mobilní telefony. Zde kraluje OS pro mobilní telefony od společnosti Google, Android. Existují, ale i další distribuce pro mobilní telefony jako je OpenMoko používaný v mobilu Neo FreeRunner nebo Bada od společnosti Samsung.

další,...

2.1.4 Architektury

Linuxové distribuce jsou také směřovány na různé procesorové architektury, většinou distribuce nesměřují jenom na jednu architekturu, ale na více architektur současně. Příkladem za všechny je distribuce Debian, která je známá také podporou pro největší množství architektur. Např. MS Windows 7 je dostupný pouze pro architektury x86 a x86_64 (x64), zatímco Debian podporuje oficiálně 10 architektur a neoficiálně 16 architektur. Nejčastěji však většina distribucí podporuje architektury x86 a x86_64 a v poslední době také ARM.

2.1.5 Další rozdíly

Rozdíly, i když ne tak zásadní, jsou v softwarové výbavě. Ta sice jde ruku v ruce s nasazením Linuxu, ale ve světě desktopů a hlavně distribuce Ubuntu se v posledních několika letech objevují distribuce, které se odlišují pouze v základním uživatelském prostředí. Oficiálně společnost Canonical (tvůrce Ubuntu) vytváří 4 distribuce, které se odlišují od Ubuntu pouze v základním uživatelském prostředí a další 4 distribuce, které se liší v softwarové výbavě. **Ubuntu, Kubuntu, Edubuntu, Xubuntu, Gobuntu, Lubuntu, Fluxbuntu, Ubuntu Netbook Edition**

Velký rozdíl zavádí do linuxových distribucí **GoboLinux**. Tato distribuce mění základní rozdělení Linuxového filesystému na více intuitivní a uživatelsky příznivější.V rootu distribuce GoboLinux lze nalézt pouze složky **Programs, Users, System, Files, Mount, Depot**. Programs je velice zajímavá složka, protože v GoboLinuxu obsahuje všechny programy a jejich soubory a díky promyšlené struktuře je možné provozovat v GoboLinuxu hned několik verzí stejného programu. Programs také zajišťuje databázi balíčkovacího systému, která je řešena pomocí složek programů. Chytrost tohoto řešení je však trochu degradována řešením zpětné kompatibility s Linuxovým standardem, která je řešena přes velké množství symbolických adres. [45] [44]

3 DISTRIBUCE LINUX FROM SCRATCH

LFS je projekt poskytující návod, který vás provede krok za krokem sestavením vlastní Linuxové distribuce ze zdrojových kódů. [7]

3.1 Užití Linux From Scratch

Spousta lidí se může divit, proč se obtěžovat se sestavováním LFS, když je možné si jednoduše stáhnout už sestavenou distribuci Linuxu. Avšak i přes některé obtíže jenž sestavování LFS skýtá, má tento proces své výhody. [3]

Obr. 2. Logo Linux From Scratch

- LFS učí uživatele, jak Linux funguje uvnitř.
- Sestavování LFS učí o všem, co v systému běží, jak jednotlivé části pracují a závisí jedna na druhé. A jak upravit systém dle potřeby.
- Sestavením LFS získáme velice kompaktní Linuxový systém
- Pokud instalujeme běžnou Linuxovou distribuci, nainstalujeme i množství programů, které pravděpodobně nikdy nepoužijeme. LFS jde sestavit i ve velikosti nepřesahující 100MB
- LFS je extrémně flexibilní. Máme moc systém sestavit přesně podle vašich potřeb.
- LFS můžeme zabezpečit na míru. Nemusíme čekat, jako u binárních distribucí, až někdo sestaví balíček s bezpečnostním patchem, který potřebujeme. Můžeme si zkompilovat SW s patchem sami.

3.2 Další projekty LFS

BLFS Beyond Linux From Scratch - projekt, jehož cílem je vytvořit návod pro rozšíření základního sestavení LFS a umožnit tak uživateli co nejjednodušeji dotvořit vlastní distribuci.

ALFS Automated Linux From Scratch - cílem projektu je vytvořit návod jakým způsobem vygenerovat kód pro automatické sestavování LFS a BLFS

CLFS Cross Linux From Scratch - jak už název napovídá, projekt umožňuje cross-compilaci, tedy sestavení LFS na mnoha jiných typech systémů a architektur.

HLFS Hardened Linux From Scratch - se snaží o vytvoření maximálně bezpečného sestavení LFS, odolného proti hrozbám hackerů a jiným bezpečnostním hrozbám.

Hints jedná se o kolekci dokumentů, které popisují jak vylepšit naše sestavení LFS jinak než je popsáno v LFS a BLFS.

LiveCD projekt, jehož hlavním cílem je vytvořit LiveCD vhodné jako hostitelský systém pro sestavení LFS.

4 SCRIPTOVÁNÍ V JAZYCE BASH

(**B**ourne **A**gain **SH**ell)
Interpretovaný, neobjektový jazyk, určený především pro administraci a automatizaci *nix operačních systémů. Skripty obvykle většinu požadované činnosti vykonávají voláním systémových utilit. Podpora syntaxe Bashe je u textových editorů obvyklá. Bash je nainstalován prakticky na každém desktopovém Linuxu a na mnoha dalších *nix systémech.[13]

4.1 Základní informace a příklady

4.1.1 Komentáře

Komentáře jsou v bashi jako v jiných jazycích značeny znakem # (sharp)

```
  # cokoliv za tímto znakem je komentář
```

Každý skript by měl začínat komentářem, který říká jaký interpret shellu se má použít, pokud toto není uvedeno, je použit defaultní interpret, což může znamenat problémy pokud jsme napsali skript např. pro bash a v systému je defaultně jiný odlehčený interpret.

```
#!/bin/bash
```

Jaký defaultní interpret je použit, můžeme zjistit pomocí příkazu:

```
$ echo "/bin/sh -> `readlink -f /bin/sh`"
/bin/sh -> /bin/bash
```

4.1.2 Výpis na terminál

K výpisu na terminál slouží příkaz ‘echo‘

```
  echo hello world
```

echo dokáže vypisovat také proměnné

```
  echo $PS1
```

více v manuálových stránkách **man echo**

4.1.3 Proměnné

Celý linuxový systém využívá proměnných a funkcí uložených přímo v BASHi. Vypsat všechny proměnné a funkce známé aktuálnímu interpretu příkazů je možné příkazem *set*[14].
Mezi základní proměnné v systému patří např.

```
$USER #uloženo uživatelské jméno
$LANG #jazyk systému
$PS1  #nastavení uživatelského promtu
$BASH #uložena adresa defaultního interpretu příkazů
```

Ukázka práce s proměnnými.

```
$ jedna="Lokální proměnná"
$ export DVA="Proměnná exportovaná do podřízeného shellu"
$ readonly TRI="Proměnná pouze pro čtení, ale jen na lokální úrovni"
$ export TRI
$ export
declare -x DVA="Proměnná exportovaná do podřízeného shellu"
declare -rx TRI="Proměnná pouze pro čtení, ale jen na lokální úrovni"
$ readonly
declare -rx TRI="Proměnná pouze pro čtení, ale jen na lokální úrovni"
$ echo $jedna
Lokální proměnná
$ TRI="Nová hodnota"
bash: TRI: readonly variable
$ bash
$ TRI="Nová hodnota"
$ echo $jedna
$ echo $DVA
Proměnná exportovaná do podřízeného shellu
$ echo $TRI
Nová hodnota
$ unset TRI
```

4.1.4 Roury a přesměrování

Opravdu důležitým prvkem jsou roury a přesměrování. Roura se značí pomocí operátoru | a připojuje výstup jednoho procesu na vstup druhého procesu.

```
  cat /var/log/auth.log | grep ssh
```

Pro přesměrování slouží operátory:

```
>  #přesměrování standardního výstupu do souboru,
    jestliže soubor existuje bude přepsán
>>  #jako předchozí, ale data přidá na konec souboru
<  #přesměrování standardního vstupu do souboru
<<text  #jako předchozí, ale při výskytu řetězce text zašle
         znak konce souboru
```

4.1.5 Deskriptor souboru

BASH rozlišuje 3 deskriptory souboru

0 standardní vstup

1 standardní výstup

2 standardní chybový výstup

```
{
 #funkce a různé procedury
} 2>&1 | tee $BUILD_DIR/LOG_$PROGRAM.log
```

4.1.6 Základní příkazy

cp kopíruje soubory

rm ruší soubory

mkdir vytváří adresáře

rmdir ruší prázdné adresáře

ln vytvoří odkazy na soubory

chmod změní přístupová práva k souborům

ls, dir, vdir vypíše obsah adresářů

find vyhledávání souborů

which zobrazí absolutní cestu k programu

df vypisuje informace o připojených FS

ps informace o spuštěných procesech

cat, less výpis souboru na obrazovku

xargs spustí zadaný příkaz a zbylé argumenty čte ze standardního vstupu

grep tiskne řádky, které odpovídají zadanému vzoru

wc vypíše počet písmen, slov a řádků

sort setřídí řádky

4.1.7 Kulaté a složené závorky

Složené závorky se používají nejen pro oddělení jména proměnné nebo pole od dalších znaků v rámci příkazu (viz. kapitola 4.3.4/s24). Dokážou také seskupovat několik příkazů do jednoho celku. To se může hodit hned v několika případech: když chceme použít konstrukci, která umí vykonat pouze jeden element, chceme-li najednou přesměrovat výstup z několika příkazů (viz. kapitola 4.1.5/s19) nebo třeba pokud chceme nechat celou skupinu úloh provést na pozadí. [2]
Lze tak bez obav zadat:

```
{ sleep 10 ; echo Budíček! ; } &
```

Středník za posledním příkazem je zde nutný, podobně jako mezera za otevírací a před zavírací závorkou. Můžeme použít také závorky kulaté, které mezery ani závěrečný středník nepotřebují. [2]
Příkaz by pak mohl vypadat takto:

```
(sleep 10 ; echo Budíček!) &
```

Vedle způsobu zápisu se konstrukce s kulatými a složenými závorkami liší ještě v jednom – předávání hodnot proměnných prostředí. To, co uzavřeme do složených závorek, poběží stále v aktuálním Bashi, takže po skončení těchto operací budou provedené změny stále platit. [2]

```
$ husy="5"
$ { husy="3" ; echo $husy ; }
> 5
$ echo $husy
> 3
```

Na první echo (to v závorkách) by měla být reakce 3, protože jsme v závorkách hodnotu proměnné změnili z 5 na 3. Na druhé echo (za závorkami) bude odpověď zase 3, protože příkazy provedené v závorkách jsou platné pro aktuální shell. Jinak tomu samozřejmě bude v případě závorek kulatých. [2]

```
$ husy="5"
$ (husy="3" ; echo $husy)
>3
$ echo $husy.
>5
```

V okamžiku otevření závorky se spustí nový shell, který provede pouze příkazy v závorce. První echo tedy opět vypíše hodnotu 3, kterou nastavujeme uvnitř závorek. Potom se závorka zavře, ukončí se příslušný Bash a s ním zmizí i hodnoty proměnných v něm nastavené. Proto druhý shell vypíše hodnotu 5, kterou jsme v něm nastavili před tím, než byl zavolán druhý Bash. [2]

4.2 Podmínky a cykly

4.2.1 IF

Obecná struktura:

```
if výraz;
  then příkazy
 elif výraz;
  then příkazy
 else příkazy
fi
```

Ukázka syntaxe:

```
if [ "$USER" == "root" ]; then
  echo "Ahoj admine";
elif [ "$USER" == "Martin" ]; then
  echo "Ahoj Martine";
else
  echo "Ahoj nějaký jiný uživateli";
fi
```

POZOR!!! za znakem [␣musí být mezera! Jedná se o program a za mezerou jsou jeho argumenty.

Místo [je možno použít *test*. Jsou to stejné programy svázané pevným odkazem. Stejná ukázka s použitím programu *test*[14]:

```
if test "$USER" == "root" ; then
  echo "Ahoj admine";
elif test "$USER" == "Martin" ; then
  echo "Ahoj Martine";
else
  echo "Ahoj nějaký jiný uživateli";
fi
```

Podmínky je samozřejmě možno spojovat pomocí operátorů && (a zároveň platí) a || (nebo platí). Operátor || má velké využití ve skriptech pro testování pokud funkce skončila úspěšně[14].

```
#testujeme dvě podmínky
if [ $USER == "root" ] && [ $LANG == "cs_CZ" ]; then
  echo "Jsi český admin"
fi
#pokud funkce skončí chybou skript skončí s příznakem 1
funkce || exit 1
```

Operátory testování výrazů:

```
[ výraz ] - délka řetězce je nenulová
[ -z výraz ] - délka řetězce je nulová
[ výraz1 == výraz2 ] - řetězce jsou shodné
[ výraz1 != výraz2 ] - řetězce jsou různé
[ výraz1 -eq výraz2 ] - čísla jsou shodná
[ výraz1 -le výraz2 ] - výraz1 <= výraz2
[ výraz1 -lt výraz2 ] - výraz1 < výraz2
[ výraz1 -ge výraz2 ] - výraz1 >= výraz2
[ výraz1 -gt výraz2 ] - výraz1 > výraz2
[ výraz1 -ne výraz2 ] - čísla jsou různé
```

Operátory testování souborů:

```
[ výraz1 -ef výraz2 ] - soubory sdílejí stejný i-uzel
[ výraz1 -nt výraz2 ] - první soubor je novější
[ výraz1 -no výraz2 ] - první soubor je starší
[ -e výraz ] - soubor existuje
[ -d výraz ] - soubor je adresář
[ -f výraz ] - soubor je obyčejný soubor
[ -L výraz ] - soubor je symbolický odkaz
[ -w výraz ] - soubor je zapisovatelný
[ -x výraz ] - soubor je spustitelný
```

4.2.2 CASE

Obecná struktura:

```
case slovo in
  vzory ) příkazy;;
esac
```

Výběr grafického prostředí pomocí case:

```
case $1 in
        gnome) exec gnome-session;;
        kde) exec openbox-session;;
        *) echo "vyber gnome nebo kde";;
   #*) znamená cokoli jiného
esac
```

4.2.3 FOR

Příkaz for funguje stejně jako v jiných programovacích jazycích, ale jeho syntaxe je trochu jiná.

```
#příkaz bude postupně do proměnné $cislo dosazovat hodnoty
#a echo je bude vypisovat na obrazovku
for cislo in 10 20 30 40 50 60 70 80 90 100; do
  echo $cislo
done
```

4.2.4 WHILE

```
cislo=0
# Podmínka je splněna jestliže $cislo != 100
while [ "$cislo" -ne 100 ]; do
  cislo=$((cislo + 10))
  echo $cislo
done
```

4.2.5 UNTIL

```
cislo=0
# Cyklus pokračuje dokud není splněna podmínka
until [ "$cislo" -eq 100 ]; do
  cislo=$((cislo + 10))
  echo $cislo
done
```

4.3 Funkce, pole, příkazy

4.3.1 Speciální proměnné

Informace o názvu skriptu, počtu předaných argumentů a argumenty samotné jsou uloženy ve speciálních proměnných. Tyto proměnné se používají stejným způsobem ve skriptech i ve funkcích[14].

$0 název skriptu

$# počet předaných argumentů

$IFS seznam znaků, který je použit k oddělování slov atp., např. když shell čte vstup

$1 až $9 první až devátý argument předaný skriptu

$n libovolný n-tý argument předaný skriptu

$* obsahuje všechny argumenty oddělené prvním znakem z $IFS

$@ jako předchozí, ale k oddělení se nepoužívá první znak z $IFS

4.3.2 Funkce

Provádění funkcí je mnohem rychlejší než provádění skriptů, protože funkce si shell udržuje trvale předzpracované v paměti. Funkce musí být definována dříve než bude použita. Příkaz export lze použít i pro funkce, ale musí být zapnutý mód allexport[14].

```
$ set -o allexport
$ prvni_funkce() {
> echo "Jsem první funkce a vypisuji text"
> }
$ export prvni_funkce
$ prvni_funkce
Jsem první funkce a vypisuji text
$ bash
$ prvni_funkce
Jsem první funkce a vypisuji text
```

Pomocí klíčového slova local můžeme také vytvořit lokální proměnné funkce. Jestliže bude existovat globální proměnná se stejným názvem, bude ve funkci potlačena[14].

```
#!/bin/bash
jedna="První globální proměnná"
dva="Druhá globální proměnná"
lokalni_promena() {
  local jedna="První lokální proměnná"
  echo $jedna
  echo $dva
}
lokalni_promena
echo $jedna
echo $dva
```

4.3.3 Příkazy

Příkazy můžeme rozdělit na zabudované a normální. Zabudované příkazy nemůžeme spustit jako externí programy, ale většinou mají své ekvivalenty ve formě externích programů. Normální příkazy jsou externí programy a jejich vykonání je pomalejší než u zabudovaných příkazů[14].

break vyskočí z cyklu

: nulový příkaz

continue spustí další iteraci cyklu

. provede příkaz v aktuálním shellu

eval vyhodnotí zadaný výraz

shift posune poziční parametry

read načte uživatelský vstup, jako argument se použije název proměnné, do které se má uložit

stty mění a vypisuje charakteristiky terminálové linky

exec spustí nový shell nebo jiný zadaný program a nebo upraví deskriptor souboru

exit n ukončení skriptu s návratovým kódem n (n = 0 - úspěšné ukončení, n = 1 až 125 - chyba, ostatní n jsou rezervovány)

printf není dostupný ve starých shellech a při vytváření formátovaného výstupu byste mu měli dávat přednost před příkazem echo podle specifikace X/Open

4.3.4 Pole

Pole se v BASHi definují do kulatých závorek a položky se oddělují mezerou. Vyvolat položky lze pak jednoduše jako jiné proměnné, ale s indexem v hranatých závorkách. [1]

```
packagename=(name dlink archivename foldername extension md5)
echo $packagename[0]
```

Ukládat položky pole do proměnných lze také velmi jednoduše:

```
promenna = ${packagename[0]}
```

5 KOMPILACE ZDROJOVÝCH KÓDŮ

5.1 Překladač (kompilátor)

Překladač (též kompilátor, anglicky compiler z to compile – sestavit, zpracovat) je v nejčastějším smyslu slova nástrojem používaným programátory pro vývoj softwaru. Kompilátor slouží pro překlad algoritmů zapsaných ve vyšším programovacím jazyce do jazyka strojového, či spíše do strojového kódu. Z širšího obecného hlediska je kompilátor stroj, respektive program, provádějící překlad z nějakého vstupního jazyka do jazyka výstupního. Z matematického hlediska je kompilátor funkce, která mapuje jeden nebo více zdrojových kódů podle překladových parametrů na kód ve výstupním jazyce. [8]

5.2 Důležité soubory

Zdrojové kódy programů jsou uloženy v souborech rozličných formátů podle účelu a jazyka, v němž je program napsán. Ale každý program by měl obsahovat i další soubory, které mají význam pro uživatele, kteří chtějí program zkompilovat nebo ho dále upravovat. Tyto soubory jsou ale spíše dobrým mravem a záleží na programátorovi jestli je jeho projekt bude obsahovat.

5.2.1 LICENSE (COPYING)

Soubor obsahující licenci pod kterou je program vydán. Licence je důležitá, protože určuje jak smí uživatel s programem nakládat. U programů s otevřeným zdrojovým kódem se nejčastěji používají licence GNU GPL a BSD.
GPL v jednoduchosti říká, že můžete s programem libovolně nakládat, avšak pokud provedete jakékoliv modifikace, musíte zdrojové kódy opět uvolnit pod stejnou licencí *(jedná se o licenci tzv. virovou)*. Anglický originál licence je lze možno nalézt na opensource.org Pod touto licencí je šířeno Linuxové jádro i sada nástrojů projektu GNU, které Linuxové jádro standardně distribuováno.
BSD Umožňuje volné šíření licencovaného obsahu, přičemž vyžaduje pouze uvedení autora a informace o licenci, spolu s upozorněním na zřeknutí se odpovědnosti za dílo. [9] Anglický originál licence je lze možno nalézt na opensource.org. Pod touto licencí šířeno např. jádro stejnojmenného operačního systému BSD nebo operační systém HAIKU.
Často bývá licence uvedena jako součást souboru README.

5.2.2 INSTALL

Soubor INSTALL obsahuje instalační informace pro uživatele. Obsahuje informace o tom, jak program zkompilovat a informace o jeho nastavení. U menších projektů bývají často tyto informace součástí souboru README, naopak u velkých projektů se často tyto informace přesunují na internet.

5.2.3 Dokumnetace

Dokumentace obsahuje informace o používání programů a je velmi důležitá především u knihoven, protože obsahuje popis funkcí, které daná knihovna poskytuje. Většina distribucí poskytuje dokumentaci v samostatných balíčcích, protože spousta uživatelů dokumentaci nijak nevyužije a ušetří se tím místo v uživatelově počítači a především se tím sníží zatížení a množství přenesených dat z repozitářů distribucí.

5.2.4 README

Soubor obsahuje základní informace o programu. Může, a často i obsahuje, informace ze všech výše uvedených souborů. Avšak někdy není vůbec, protože je vše uvedeno ve výše uvedených souborech.

5.2.5 Configure

Soubor s nastaveními pro překlad. Jedná se v zásadě o BASH skript, kterým můžete nastavit co a jak se má v programu kompilovat. Různými volbami, které jsou uvedeny v INSTALL,README, nebo v nějaké online dokumentaci, můžete nastavit *např. cesty kam se má program nebo jeho části nainstalovat, nebo také které části programu se mají přeložit.* Configure je skriptem BASHe a proto je také tak volána, výsledkem configure je vygenerovaní souboru, které řídí překlad, nejčastěji Makefil(u).

5.2.6 Další soubory a alternativy některých souborů

5.3 Kompilace

Samotná kompilace sestává z prostudování nejčastěji souborů INSTALL nebo README a postupem podle návodu v nich. Ale obecně ji lze zahrnout do posloupnosti tří "příkazů".

```
./configure [options]
```

vygeneruje soubory potřebné pro překlad

```
make
```

zkompiluje zdrojové kódy [1)]

```
make install
```

nahraje soubory do systému všechny potřebné soubory

5.4 Možnosti tvorby distribuce

Linuxovou distribuci lze vytvořit více způsoby, nejběžnějším způsobem je sestavení ze zdrojových kódů. Je třeba dávat pozor na licenční ujednání a dle GNU/GPL licence (pod kterou je distribuováno linuxové jádro) musí autor uvolnit zdrojové kódy. U linuxového jádra to zahrnuje většinou použité patche.

5.4.1 Vytvoření ze zdrojových kódů

Vytvoření ze zdrojových kódů reprezentuje například použitá "distribuce"Linux From Scratch. Postup se obecně skládá z několika částí.

1. Nejdříve je třeba oddělit systém a vytvořit toolchain, který slouží k sestavení nové distribuce nezávislé na předchozím hostovskému systému.
2. Sestavení vlastní sady základních nástrojů pro použití systému. Sada obsahuje překladače, nástroje pro práci se soubory a jiné potřebné programové vybavení.
3. Nastavení bootovacího procesu systému.
4. Vytvoření tabulky souborových systémů, kompilace jádra systému, nastavení bootloaderu

[1)] make neslouží pouze ke kompilaci zdrojových kódů programů, ale i jako univerzální utilita pro překlad projektů. LFS jej například používá pro generování PDF,HTML,... verze knihy z XML souborů

5.4.2 Sestavení ze stávající linuxové distribuce

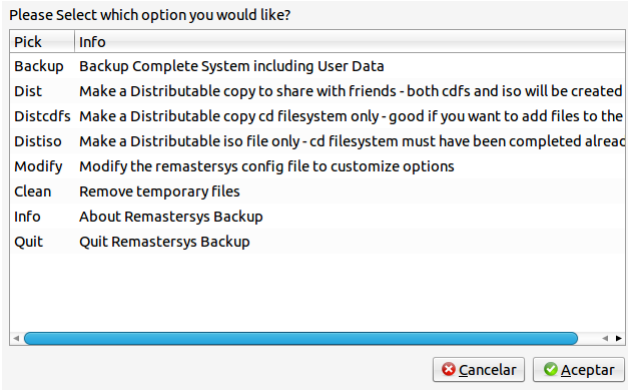
Existuje hned několik nástrojů na sestavení tzv. derivátu Linuxové distribuce. Většinou jsou používány pro sestavení instalačního ISO obrazu na míru potřeb uživatele. To zahrnuje balíčky použité při instalaci, různé nastavení, grafická témata, písma, apd,... Populární se staly především deriváty distribuce Ubuntu, pro Ubuntu existuje i nejvíce těchto nástrojů. K dispozici jsou nástroje jak pro desktop s běžným GUI nebo CLI,tak i nástroje webové, které sestaví distribuci v cloudu a uživatel stáhne jenom hotový obraz.

Remastersys

Remastersys je jednoduchý grafický nástroj určený pro Ubuntu, Debian a všechny jejich deriváty. Remastersys dokáže převést aktuální běžící distribuci na vašem počítači do ISO obrazu. Je výborným nástrojem pro zálohování dat systému, protože uloží všechny vaše uživatelská data a nastavení.[15]

Výpis možností použití Remastersys: Jedná se opravdu o jednoduchý program.

```
$ remastersys
Usage of remastersys 2.0.18-1 is as follows:
   sudo remastersys backup|clean|dist [cdfs|iso] [filename.iso]
Examples:
   sudo remastersys backup custom.iso
    (to make a livecd/dvd backup and call the iso custom.iso)
   sudo remastersys clean
    (to clean up temporary files of remastersys)
   sudo remastersys dist
    (to make a distributable livecd/dvd of your system)
   sudo remastersys dist cdfs
    (to make a distributable livecd/dvd filesystem only)
   sudo remastersys dist iso custom.iso
    (to make a distributable iso named custom.iso but only
                              if the cdfs is already present)
```



Obr. 3. Grafické rozhraní programu Remastersys

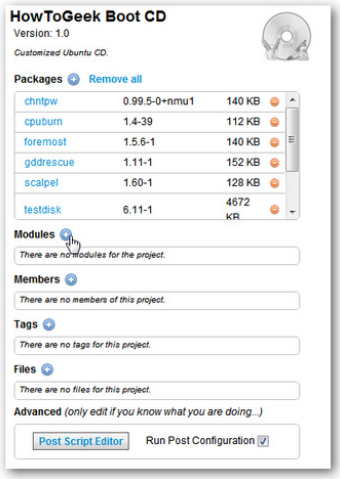
The Ubuntu Customisation Kit (UCK)

Program je určen pro Ubuntu a všechny jeho deriváty. Dovoluje odebrat nebo přidat jakékoliv aplikace na instalačním CD. UCK je určeno spíše pokročilejším uživatelům, protože po vytvoření LiveCD do něj přihlásí uživatele prostřednictvím chroot a dovolí

mu tak změnit prakticky cokoliv. Aplikace má jednoduché grafické rozhraní se spoustou dialogů, které nás provedou vytvořením LiveCD.[15]

Reconstructor

Jedinou placenou aplikací je program Reconstructor. Je určen pro vytvoření vlastního instalačního CD ze stávajícího ISO obrazu. Umožňuje změnit wallpaper, témata, icony, aplikace a mnoho jiného. Jedná se o webovou aplikaci a její cena je $5.[15]



Obr. 4. Webové rozhraní Reconstructoru

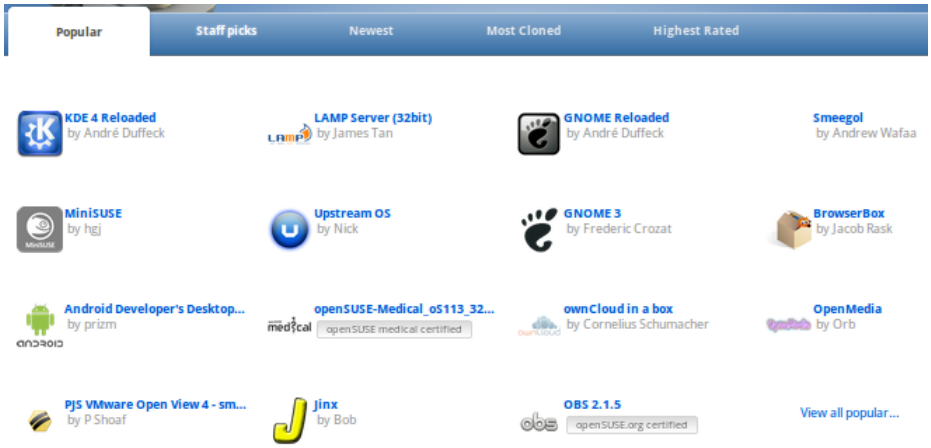
Revisor

Revisor je aplikace pro vytvoření vlastního instalačního CD/DVD/USB založeném na distribuci Fedora. Má jak grafické rozhraní, tak CLI. Další rozdíl je také to, že Revisor nevytváří instalační medium z existujícího obrazu, ale stahuje vše z internetu, takže rychlost vytvoření média je také závislá na rychlosti vašeho připojení.[15]

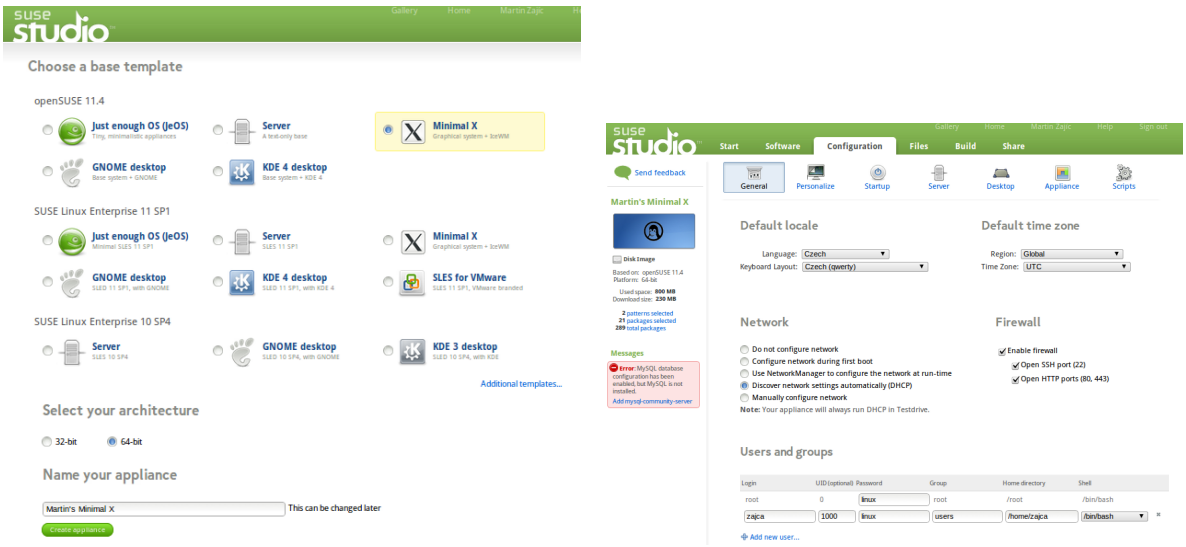
Obr. 5. Grafické rozhraní programu Revisor

SUSE Studio

Asi nejzajímavějším programem je webová aplikace od společnosti Novell, SUSE Studio. Nejenomže umí nakonfigurovat velké množství nastavení. Ale velmi dobrou vlastností je testování systému přímo v prohlížeči. Výborná je také galerie, která obsahuje obrazy vytvořené ostatními uživateli, lze tak bez práce najít už funkční obraz dle potřeb uživatele, a případně si již vytvořený obraz upravit.



Obr. 6. Webové stránka s výběrem již existujících sestavení



Obr. 7. Webové rozhraní s nastavením systému, vlevo výběr základního rozhraní, vpravo podrobnější nastavení

Pungi

Pungi je nástrojem, který používají vývojáři Fedory pro sestavení oficiálních vydání. Jedná se o program bez grafického rozhraní napsaný v pythonu a stejně jako Revisor stahuje balíčky z internetu a sestavuje z nich instalační médium.[15]

Builder

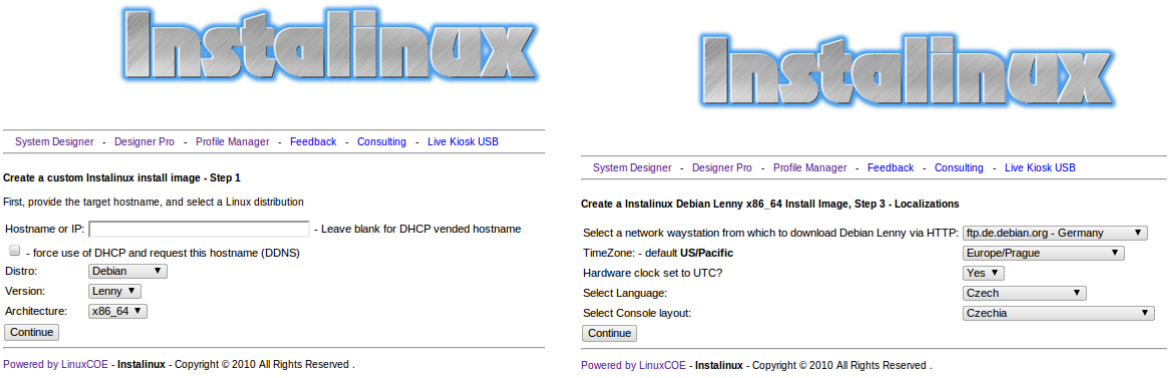
Builder je série skriptů napsaných v BASHi, které používají vývojáři derivátu distribuce Ubuntu "gNewSense"na sestavení jejich distribuce. Ke skriptům je také napsán přehledný manuál, jak je používat, který je dostupný online [15] [50].

MySlax Creator

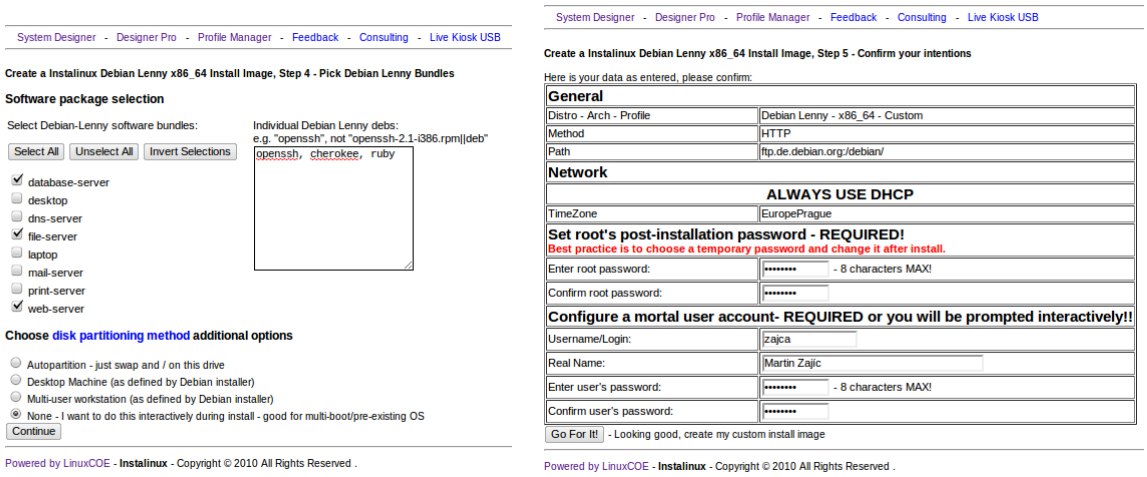
MySlax Creator jsou skripty určené k úpravě distribuce Slackware, především tedy k vytvoření vlastního obrazu odvozené distribuce SLAX. Jejich odlišnost je v tom, že fungují přímo z Windows.

Instalinux

Jednoduchým, ale mocným nástrojem je Instalinux, umožňuje uživateli vybrat z několika distribucí (Debian, Ubuntu, CentOS, Fedora, OpenSUSE, Scientific) a v dalších krocích základní nastavení systému a výběr aplikací. Výsledkem je ISO obraz, kterým nemusí mít ani 30MB. Pro podrobnější nastavení existuje Designer Pro, který umožňuje nastavit mnohem více. Výbornou vlastností je uložení profilu, který je možné použít znovu a upravit si jej.



Obr. 8. Webové rozhraní Instalinux, vlevo výběr distribuce, vpravo nastavení jazyka a časové zóny



Obr. 9. Webové rozhraní Instalinux, vlevo výběr aplikací, vpravo nastavení uživatelů

Takto vytvořená distribuce může mít i 8MB!

6 LIVECD

Ač by se mohlo zdát, že liveCD je nějaká velmi složitá věc k vytvoření, není to tak úplně pravda. Nejjednodušší forma LiveCD je taková, že vložíme CD do mechaniky, ten nabootuje a spustí se příkazový řádek, ve kterém je možné pracovat. Takto fungovaly a často ještě fungují instalační CD různých distribucí. Dnes se velmi často používá několik druhů instalačních/live CD.

- Instalační CD s příkazovou řádkou a instalací pomocí textového nástroje (Archlinux,...)
- Instalační CD s grafickým instalátorem (Debian,...)
- LiveCD, ve kterém lze z grafického Live prostředí spustit instalátor (ve většině případů lze spustit grafický isntalátor i bez grafického live prostředí jako v předchozí položce) (Ubuntu,...)
- Čistě LiveCD, takový disk slouží pouze pro vyzkoušení většinou grafického prostředí nebo k opravám stávajícího systému (neobsahuje instalátor) (GNOME Live Media, Gparted,...)
- LiveCD určené pro běžnou práci. Takové LiveCD umí např. nahrát systém do RAM, pak už není třeba číst z CD nebo také pokud se jedná o LiveUSB, je možné uložit změny v systému na disk a ty jsou načteny při dalším spuštění. (Slax,Chakra,...)

6.1 Výhody/nevýhody LiveCD

Vyhodou LiveCD je především možnost **vyzkoušet si celý operační systém bez nutnosti instalace**. Kompromisem LiveCD je pak **rychlost**, která se odvíjí od rychlosti čtení CD/DVD mechaniky. Rychlost LiveCD je možné zvýšit použitím jiného média než CD nebo DVD, např. použitím FlashDisku. Rychlost běhu lze zvýšit také **nahráním celého systému do paměti RAM**, tento proces má však velkou nevýhodu v délce trvání, protože celé LiveCD musí být přečteno a zapsáno do paměti RAM.

Další nevýhodou je, až na výjimky, nemožnost uložení změn v systému zpět na přenosné médium.
Samozřejmě LiveCD není většinou používáno k běžně práci, jeho výhodou je vyzkoušení neznámého operačního systému nebo grafického prostředí bez nutnosti mnohdy zdlouhavé instalace, což převyšuje nad nevýhodami.

6.2 Možnosti tvorby LiveCD

Vytvořit LiveCD lze několika způsoby, nejjednodušší je způsob úpravy stávajícího LiveCD, jak bylo i popsáno (viz. kapitola 5.4.2/s27).

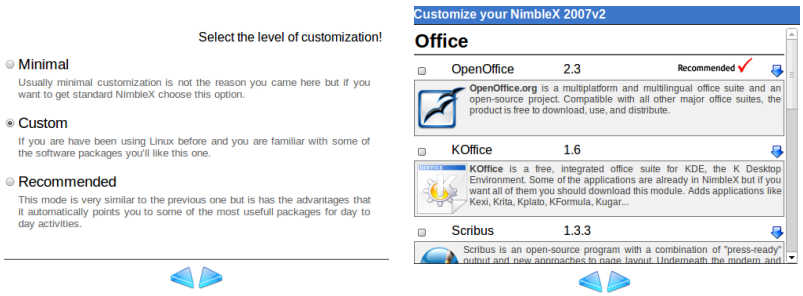
6.2.1 Obecný postup tvorby LiveCD

Obecný postup tvorby LiveCD se dá shrnout do několika kroků.

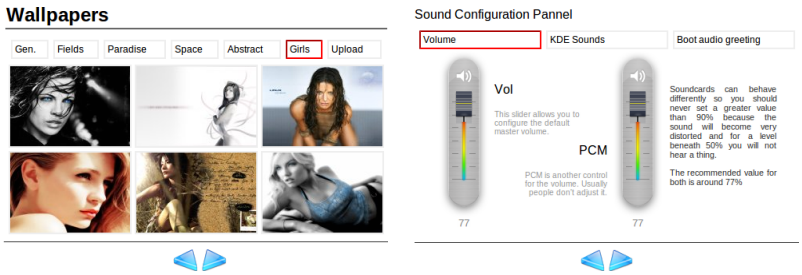
1. Vytvoření adresářové struktury LiveCD
2. Připojení Virtual File System (VFS), ten zahrnuje složky: /sys /proc /dev /dev/shm /dev/pts /var/run
3. Nainstalování základních balíčků
4. Nainstalování modulů squashfs a aufs
5. Nainstalování kernelu a vytvoření initramfs pomocí mkinitcpio s minimálně aktivovanými moduly aufs, squashfs a loop, které jsou potřebně pro běh LiveCD.
6. Nainstalování a nastavení bootloaderu (GRUB)
7. Doinstalování dalších balíčků
8. Komprimace a vytvoření squashfs souboru pomocí programu mksquashfs
9. Vytvoření iso obrazu pomocí programu genisoimage, který vytvoří i zaváděcí tabulku disku.

6.2.2 NimbleX

Dalším nástrojem je jednoduchá webová aplikace na vytvoření vlastního derivátu NimbleX. Umožňuje vytvořit vlastní LiveCD obraz několik jednoduchými kroky, na výběr je stáhnutí minimálního obrazu nebo sestavení vlastního obrazu. U vlastního obrazu si uživatel vybere aplikace přehledně rozdělené do kategorií, wallpaper, nastavení zvuku, nastavení uživatelů a jazyk systému. Pak už jen počká na vygenerování obrazu, který je následujících 12hodin ke stažení.



Obr. 10. Webové rozhraní s nastavením NimbleX, vlevo výběr míry nastavení, vpravo nastavení programů v sekci office



Obr. 11. Webové rozhraní s nastavením NimbleX, vlevo výběr wallpaperu, vpravo nastavení zvuku

6.2.3 Linux-Live

Linux-Live je sada skriptů, která není závislá na distribuci, ale lze ji použít na jakémkoliv Linuxu. Prakticky stačí stáhnou skripty ze stránek Linux-Live a zkompilovat si kernel s podporou aufs a squashfs nebo si stáhnout již zkompilovaný kernel ze stránek Linux-Live. Skripty už pak samy vytvoří Live systém, který je možné nabootovat z CD nebo USB.

6.2.4 Další možnosti tvorby LiveCD

Linux-Live je asi nejčastějším způsobem vytvoření LiveCD. Spousta distribucí má ale vlastní nástroje jako: The Ubuntu Customisation Kit (UCK) nebo Reconstructor, které již byly popsány (viz. kapitola 5.4.2/s27). Pro fedoru např. exituje nástroj **livecd-creator** pro tvorbu Fedora LiveCD a podrobný návod je dostupný na stránkách fedory [41]. Pro debian existuje **Debian Live Project**, který poskytuje webové rozhraní podobné nástroji Instalinux (viz. kapitola 5.4.2/s30) a je zdarma dostupný na internetu. [42] Další zajímavé nástroje pro tvorbu LiveCD existují např. pro Archlinux. Jsou hned tři a všechny jsou popsány na wiki stránkách distribuce Archlinux [43]. První je oficiální sada nástrojů, používaná k tvorbě oficiálních vydání disků nazvaná **Archiso**. Druhým nástrojem je **larch**, který nepotřebuje k sestavení jako hostitelský systém Archlinux, ale funguje v jakémkoliv Linuxu, protože si potřebné aplikace stáhne z repozitářů Archlinuxu. Posledním nástrojem je **poison-livecd-creator**, jedná se o opravdu jednoduchý Makefile a několika jednoduchými kroky lze docílit vytvoření LiveCD na bázi Archlinuxu.

7 ZABEZPEČENÍ LINUXOVÉHO SYSTÉMU

Zabezpečení OS je velice rozsáhlým a komplexním odvětvím, které se navíc velmi rychle vyvíjí a rozšiřuje. Jeho plné popsání by vydalo na několik desítek bakalářských prací, proto jen základní informace o vnitřním zabezpečení systému, nástrojích, jež toto zabezpečení rozšiřují a také o zabezpečení proti vnějšímu útoku a bezpečném přenosu informací.

7.1 Bezpečnostní model Linuxu

Standardní bezpečnostní model, vycházející z Discretionary Access Control (DAC) (viz. kapitola 7.2.1/s34), který nalezneme na většině UNIXových operačních systemů (tedy i Linuxu) je postaven na několika bitových maskách (specifikující u souboru práva pro vlastníka, skupinu, ostatní (drwxrwxrwx) a popř. nastavující některé speciální bity, jako je SUID (secure user identification number) či SGID (secure group identification number)). Další velmi typickou stránkou tohoto bezpečnostního modelu je uživatel root – administrátor a uživatel s nejvyššími možnými privilegii (jinak řečeno

uživatel, jenž může vše). [52]

Tento bezpečnostní model se v praxi v různých UNIXových klonech osvědčil a používá se dodnes. Ale jsou případy kdy není zrovna nejvýhodnější. Prvním problémem jsou ona již zmíněná přístupová práva k souborům, která jsou ze své podstaty poměrně omezená a nastavení práv tak často nutí administrátora k různým krokům, jako je např. vytváření velkého množství uživatelských skupin (a v některých případech ani to není uspokojivým řešením). Dalším kamenem úrazu je uživatel root, kterýžto jako administrátor prakticky má veškerá možná práva. Některý software část těchto práv ovšem vyžaduje – typickým příkladem je většina síťových démonů, které poslouchají na privilegovaných síťových portech (<1024). A mimo oněch potřebných práv tak program zcela zbytečně dostane plnou kontrolu nad systémem, což může být zneužito potenciálním útočníkem, pokud se v onom softwaru vyskytuje nějaká bezpečnostní chyba. Standardně se tento problém řeší tak, že program zahazuje EUID 0 (effective user identification number) v momentě, kdy již nepotřebuje speciální práva. [52]

Dalším souvisejícím problémem je nemožnost efektivní distribuce práv k systému mezi více administrátorů. Příkladem budiž situace, v níž potřebujeme svěřit administraci DNS serveru jinému člověku než tomu, který se stará o zbytek administrace systému. Nasnadě je povolení zápisu do konfiguračních souborů onoho daemonu, ale tím vyřešíme jen část problémů. Onen administrátor též musí mít možnost např. onen server restartovat či mu alespoň poslat nějaký signál (např. pro načtení nové konfigurace), což s sebou přináší nutnost superuživatelských práv. Jistým řešením je sudo či podobný nástroj, který umožňuje administrátorovi nastavit, jaké příkazy (a pod jakým uživatelem) může daný uživatel spouštět ze svého normálního uživatelského účtu. Nicméně nejedná se o dostatečně flexibilní řešení. [52]

Řešení se nabízí pomocí bezpečnostních patchů linuxového jádra implementující jiné modely přístupu než je v Linuxu standardní Discretionary Access Control (DAC).

7.2 Access Control

Kontrola přístupu uvnitř systému je jedno z nejdůležitějších zabezpečení v OS vůbec. Určuje, který proces nebo uživatel má k čemu oprávnění. Existuje několik obecných modelů tohoto zabezpečení. V Linuxu se konkrétně využívá modelu **Discretionary Access Control (DAC)**, který je standardní součástí Linuxového jádra.

7.2.1 DAC vs. MAC vs. RBAC

Bezpečnostní model využívaný většinou mainstreamových operačních systému je založen na **Discretionary Access Control (DAC)**, který je založen na bezpečnosti vlastnictvím. Pokud uživatel vlastní soubor, může nastavovat práva pro čtení, zápis a spouštění (r+w+x), pro tento soubor. V tomto modelu uživatel kontroluje data dle vlastního uvážení. Vlastník systému[1)] nemá totální kontrolu nad systémem, tu má uživatel. [21]

V Mandatory (česky Povinné) Access Control (MAC), uživatelům jsou dána povolení ke zdrojům správcem. Pouze správce může udělit oprávnění nebo právo k objektům a zdrojům. Přístup ke zdrojům je založený na úrovni zabezpečení objektu a uživateli je udělena bezpečnostní prověrka pro přístup k objektu. Pouze správci mohou upravovat objektu bezpečnostní štítek nebo uživateli bezpečnostní prověrky.

V Role-Based Access Control (RBAC), je přístup ke zdrojům založen na „roli kterou uživatel hraje“. V tomto modelu, správce přiřadí uživateli roli, která má předem přidělena určitá práva a výsady. Vzhledem k uživateli přidělené roli, může uživatel přistupovat k určitým zdrojům a plnit specifické úkoly. RBAC je také známý jako

[1)] Vlastník systému (anglicky system owner nebo také administrátor) je určen při instalaci OS, oproti účtu uživatele root nemá právo měnit např. práva systémových souborů, ale má právo měnit přístupová práva široké škály uživatelů a uživatelé nemají možnost tato ustanovení měnit.[22]

Non-Discretionary Access Control. Role přiřazené uživatelům, jsou centrálně spravovány.

7.2.2 Security-Enhanced Linux, SElinux

SELinux je bezpečnostní vylepšení pro Linux, které dává uživatelům a správcům větší kontrolu nad tím, který uživatele nebo aplikace může přistupovat k jakým prostředkům. Standardní Linuxová kontrola přístupu jako je oprávnění pro soubor (-rwxr-xr-x) je modifikovatelné uživatelem nebo aplikací, kterou uživatel spustí. Oproti tomu, kontrola přístupu v SELinuxu je určena pravidly v systému a neopatrný uživatel nebo nesprávná aplikace je nemůže nijak změnit.

SELinux také přidává mnoho možností pro řízení přístupu. Místo klasického určení, kdo může číst, psát nebo spustit soubor, např. SELinux umožňuje určit, kdo může vytvářet a rušit odkazy, přesouvat soubory a mnoho dalšího. SELinux umožňuje určit přístup k mnoha jiným zdrojům než jen k souborům, např. přístup k síťovým zdrojům nebo k mezi procesové komunikaci (IPC).[28]

7.2.3 AppArmor

AppArmor je efektivní a snadno použitelná Linuxová bezpečnostní aplikace. AppArmor aktivně chrání operační systém a aplikace před vnějšími nebo vnitřními bezpečnostními hrozbami a díky prosazování dobrého chování a prevence od neznámých, nebezpečných nebo vadných aplikací dokáže chránit systém i od tzv. zero-day útoků[2)]. V bezpečnostních pravidlech AppArmor lze definovat, k jakým systémovým zdrojům můžou jednotlivé aplikace přistupovat a jak s nimi mohou nakládat. Mnoho výchozích bezpečnostních pravidel je součástí AppArmor a pomocí kombinace vyspělé statické analýzy a learning-based nástrojů, je možné bezpečnostní pravidla AppArmor i pro velmi složité aplikace úspěšně nasadit během několika hodin.[29]

AppArmor se především oproti SElinuxu vyznačuje mnohem jednodušším nastavením.

7.2.4 TOMOYO

Tomoyo Linux je Mandatory Access Control (MAC) implementace pro Linux, díky které lze zvýšit zabezpečení systému a nebo může být pouze nástrojem systémové analýzy. Jeho vývoj byl zahájen v březnu 2003 a je sponzorován NTT Data Corporation, Japonsko.

Tomoyo Linux se zaměřuje na chování systému. Každý proces je vytvořen za dosažením určitého cíle a jako imigrační pracovník, Tomoyo Linux umožňuje každému procesu určit si zdroje potřebné k dosažení svého cíle. Pokud je povolena ochrana, Tomoyo Linux funguje jako hlídací pes a omezuje každý proces tak, jak má povoleno správcem.[30]

Mezi hlavní přednosti Tomoyo Linux patří:

- Systémová analýza
- Zvýšená bezpečnost díky Mandatory Access Control (MAC)
- Nástroje na podporu generování bezpečnostních pravidel
- Jednoduché syntaxe
- Snadné použití
- Velmi málo závislostí
- Nevyžaduje žádné úpravy stávajících spustitelných souborů.

[2)] zero-day atack je útok na vývojářům neznámou zranitelnost v systému. Zero-day je také nazýván, protože se objeví dříve než vývojáři dokáží chybu záplatovat.

7.2.5 Grsecurity

Grsecurity je inovativní přístup k zabezpečení, využívající multi-layered (vícevrstvou) detekci, prevenci a omezení modelu. Je vyvíjen pod licencí GNU GPL. [32]
Nabízí mnoho funkcí:

- inteligentní a robustní Role-Based Access Control (RBAC) systém, který může generovat bezpečnostní pravidla pro celý systém bez jakékoliv konfigurace
- Rozsáhlé kontroly
- Prevence nežádoucího kódu, bez ohledu na použité techniky (stack smashing, heap corruption, atd…)
- Prevence spuštění libovolného kódu v jádře
- Randomizaci zásobníku a knihoven
- Ochrana proti využitelným chybám null-pointer dereference [31] v jádře
- Snížení rizika úniku citlivých informací díky chybám v jádru systému
- Omezení, která umožňují uživateli zobrazit pouze své procesy
- Výstrahy zabezpečení a audity, které obsahují IP adresu osoby, která způsobila záznam

7.3 Hodnocení bezpečnosti OS a SW

Tak jako ve spoustě jiných odvětvích se i v bezpečnosti určují standardy a udělují certifikace bezpečnosti.

7.3.1 Trusted Computer System Evaluation Criteria, TCSEC

Asi nejznámější kniha popisující požadavky pro ”důvěryhodný”operační systém, vydaná v roce 1985 americkým ministerstvem obrany (Department of Defense, DoD), je známa také jako oranžová kniha (Orange book) pro svůj oranžový přebal. Dnes se tyto hodnotící kritéria již nepoužívají, původně měla být nahrazena Federal Criteries (FC), ale tento soubor kritérií nebyl nikdy nasazen a zůstal pouze ve fázi draft[3)], protože jej v roce 2005 nahradil soubor Common Criteria (CC) (viz. kapitola 7.3.4/s38)

TCSEC, hodnotí míru splnění požadavků ve třech oblastech informační bezpečnosti [23]

Zásady (Policy) metody řízení přístupu, označení stupně utajení

Odpovědnost (Accountability) zjištění identity uživatele, monitoring činnosti uživatele

Záruky (Assurance) požadavky na nezávislé hodnocení a průběžné provádění bezpečnostních funkcí

Seznam tříd bezpečnosti operačního systému podle TCSEC:

Třída D minimální ochrana - systémy nevyhovující vyšším třídám

Třída C výběrová ochrana (Discretionary Protection) prosazuje výběrové řízení přístupu

[3)] **Draft** se používá jako označení pro návrh

Podtřída C1 ochrana s výběrovým přístupem - základní oddělení uživatelů a dat chrání citlivá data před náhodným zneužitím

Podtřída C2 ochrana s řízeným přístupem - detailnější řízení přístupu a audit činností

Třída B povinná ochrana (Mandatory Protection)

Podtřída B1 ochrana bezpečnosti návěštím - přidělení stupně utajení (neformální model)

Podtřída B2 strukturovaná ochrana - existence polo-formálního modelu, oddělení citlivých a necitlivých částí systému (odolnost proti napadení)

Podtřída B3 bezpečnostní domény - modulární architektura systému (vysoká odolnost proti napadení)

Třída A verifikovaná ochrana (Verified Protection)

Podtřída A1 verifikovaný návrh, funkčně stejně jako B3, rozšíření míry formální specifikace a verifikace při návrhu a testování

Systémy třídy A se mezi komerčně dostupnými produkty nevyskytují.
Systémy s dostatečnou bezpečností pro běžný provoz se považují systémy minimálně třídy C2. [23]
Většina Linuxových distribucí dosahuje Třídy C2. [24]

7.3.2 IT Security Evaluation Criteria, ITSEC

Evropská kritéria vytvořená v roce 1990 dnes již nahrazena kritérii Common Criteries (CC) (viz. kapitola 7.3.4/s38).
Rozdělení požadavků kladených na

- Předmět hodnocení
- Míru záruk a funkčnost

Třídy záruk za správnost (E0-E6) a efektivnost odolat možným útokům [23]

E0 požadavky ITSEC nesplněny

E1 požadavek na neformální zadání bezpečnosti a popis návrhu a důkazní testy

E2 požadavek na neformální popis detailního návrhu, nezávislé testy

E3 jako třída E2 a hodnocení zdrojového kódu (SW) nebo obvodových schémat (HW)

E4 formální model bezpečnostního návrhu a provedení analýzy zranitelnosti

E5 požadavek na specifikaci návaznosti detailního návrhu a zdrojových kódů (schémat)

E6 jako třída E5 a formální popis návrhu a doložení jeho konzistence s matematickým modelem.

Třídy funkčnosti [23]

F-C1 až F-B3 podle TCSEC

F-IN vysoké nároky na integritu

F-AV vysoké nároky na dostupnost

F-DI vysoké nároky na integritu při přenosu

F-DC vysoké nároky na důvěrnost při přenosu

F-DX vysoké nároky na důvěrnost a integritu při přenosu

7.3.3 Canadian Trusted Computer Product Evaluation Criteria, CTCPEC

Kanadský soubor kriterií, který je dnes již také nahrazen Common Criteries (CC) (viz. kapitola 7.3.4/s38).

Hodnocený systém je chápán jako soubor bezpečnostních funkcí s úrovní záruk [23]

Funkce - rozdělení do skupin, v každé skupině pro každou funkci je stanoveno několik úrovní [23]

Skupina C - "Důvěrnost" 4 funkce, ochrana proti úniku informace

Skupina I - "Integrita" 7 funkcí pro udržení konsistence dat a systému

Skupina A - "Dostupnost" 4 funkce pro udržení dosažitelnosti služeb

Skupina W - "Odpovědnost" 3 funkce pro zajištění odpovědnosti uživatelů

7.3.4 Common Criteria for Information Technology Security Evaluation, CC

V České republice je dobře známa norma ISO/IEC 15408-1:1999, která je totožná s textem, zveřejněným Organizacemi sponzorujícími projekt Společná kritéria, pod názvem "Common Criteria for Information Technology Security Evaluation", verze 2.1. Tato kritéria jsou obvykle nazývána pouze "Common Criteria" a pro jejich označení se hojně používá zkratky "CC". Zkratka "CC" je ponechána i v českém překladu normy. [25]

V současné době se používá verze 3.0.

CC vznikla na základě již dříve používaných kritérií hodnocení, zejména amerických TCSEC (viz. kapitola 7.3.1/s36) a Federal Criteria, evropských ITSEC (viz. kapitola 7.3.2/s37) a kanadských CTCPEC (viz. kapitola 7.3.3/s38). Na vývoji CC se podílely národní organizace, působící v oblasti bezpečnosti a standardizace, z šesti států světa, jmenovitě Kanady, Francie, Německa, Holandska, Velké Británie a Spojených států amerických a jsou posledním výsledkem úsilí o vytvoření společného standardu v oblasti hodnocení bezpečnosti informačních technologií. V uvedených státech lze nalézt většinu z uznávaných laboratoří, které prováděly hodnocení bezpečnosti produktů v oblasti informačních technologií podle dříve široce akceptovaných kritérií TCSEC nebo ITSEC a v současné době již přešly výhradně na hodnocení podle CC. [25]

Jako formální základ pro vzájemné uznávání hodnocení uzavřely Kanada, Francie, Německo, Velká Británie a Spojené státy americké v roce 1998 dohodu "Arrangement on the Recognition of Common Criteria Certificates in the field of Information Technology Security", zkráceně nazývanou CCRA (CC Recognition Arrangement). [25]

CC jsou rozdělena do 3 částí:

Část 1/ Part 1 Úvod a všeobecný model (Introduction and general model)

Část 2/ Part 2 Bezpečnostní funkční požadavky (Security functional requirements)

Část 3/ Part 3 Požadavky na záruky bezpečnosti (Security assurance requirements)

V **první části** jsou uvedeny definice pojmů, je vysvětlena základní filosofie CC a je prezentován obecný model hodnocení. Důležitou součástí je vymezení několika stavebních prvků, které slouží pro jednotné vyjádření bezpečnostních požadavků, ať již funkčních nebo na záruku. [25]

Jsou to:

prvek (element) jako dále nedělitelný bezpečnostní požadavek ověřitelný při hodnocení

komponenta (component) jako nejmenší množina prvků pro zahrnutí do vyšších struktur v CC

rodina (family) jako seskupení komponent, z nichž všechny slouží k naplnění téhož cíle, ale liší se přísností požadavků

třída (class) jako seskupení rodin, které pokrývají jednotlivé dílčí cíle a dohromady tvoří konsistentní celek pro dosažení určitého cíle celkového

Důležitým pojmem v CC je **"profil ochrany" (Protection Profile, PP)**, který představuje implementačně nezávislou množinu bezpečnostních požadavků pro zajištění definovaných cílů. Tyto požadavky mohou být vybrány z CC nebo být vyjádřeny explicitně a mají zahrnovat i míru záruky hodnocení (Evaluation Assurance Level, EAL). PP se obvykle vytváří tak, aby byl opakovaně použitelný a musí obsahovat i zdůvodnění bezpečnostních cílů a požadavků. [25]

Pokud jde o specifický produkt nebo systém, nazývaný **"předmět hodnocení" (Target of Evaluation, TOE)**, vyjadřují se bezpečnostní požadavky, které jsou v něm realizovány, jako tzv. **"bezpečnostní cíl" (Security Target, ST)**. ST je množinou bezpečnostních požadavků, vyjádřených odkazem na PP, na existující balíky, na komponenty CC nebo explicitně. [25]

Ve **druhé části** jsou stanoveny funkční komponenty, které budou používány jako standardní způsob vyjadřování funkčních požadavků. Jde v podstatě o katalog funkčních komponent, rodin a tříd. [25]
Je definováno 11 funkčních tříd, jejichž originální anglické názvy/české názvy jsou:

- Audit/Bezpečnostní audit
- Identification and Authentication/Identifikace a autentizace
- Resource Utilisation/Využití zdrojů
- Cryptographic Support/Kryptografická podpora
- Security Management/Správa bezpečnosti
- TOE Access/Přístup k TOE
- Communications/Komunikace
- Privacy/Soukromí
- Trusted Path/Channels / Důvěryhodná cesta/kanály
- User Data Protection/Ochrana uživatelských dat
- Protection of the TOE Security Functions/Ochrana TSF

Třetí část zahrnuje komponenty pro popis požadavků na záruky. Je katalogem komponent záruk, jejich rodin a tříd. Rovněž jsou v ní definována kritéria pro hodnocení profilů ochrany (PP) a bezpečnostních cílů (ST). V CC je pro oblast záruk definováno 8 tříd, jejichž originální anglické názvy/české názvy jsou [25]:

- Configuration Management/Správa konfigurace
- Guidance Documents/Průvodní dokumentace
- Vulnerability Assessment/Posouzení zranitelnosti
- Delivery and Operation/Dodání a provoz

- Life Cycle Support/Podpora životního cyklu
- Assurance Maintenance/Údržba záruky
- Development/Vývoj
- Tests/Testy

Důležité je stanovení předdefinované vzestupné stupnice pro úroveň hodnocení. CC poskytuje 7 předdefinovaných balíků pro záruky (assurance package), známých jako Evaluation Assurance Levels (EAL), v českém překladu míry záruky hodnocení. Jsou dobře promyšlené a vyvážené, obecně aplikovatelné. Analogii lze nalézt v třídách E1 až E6 v kritériích ITSEC. Veškerá hodnocení IT podle CC se dnes provádějí na úrovni některé z EAL, převážně do úrovně EAL4. [25]

Stručně lze jednotlivé úrovně popsat následujícím způsobem:

EAL1 je vhodná, pokud je vyžadována určitá základní důvěra ve správnost fungování hodnoceného PP, ST nebo TOE, avšak hrozby nejsou považovány za vážné. Důvěry se dosahuje nezávislým testováním shody hodnoceného PP, ST nebo TOE s neformální funkční specifikací a zkoumáním předložených příruček pro uživatele. [25]

EAL2 již vyžaduje spolupráci vývojáře, který musí v podstatě dodat funkční specifikace, určité informace o návrhu bezpečnostních funkcí (na úrovni globálního návrhu, high-level design) a výsledky testování, avšak vývoj si nevyžaduje více úsilí nežli je potřebné pro dodržování dobré komerční praxe, a v podstatě nepřináší zvýšení nákladů. Poskytuje nízkou až střední nezávisle ověřenou bezpečnost v případě, že není dostupná kompletní informace z fáze vývoje. Důvěry se dosahuje analýzou vyžadované dokumentace, ověřením výsledků některých testů, analýzou síly funkcí a analýzou zřejmých zranitelností. Pro TOE musí být sestaven seznam konfigurace a vypracovány procedury pro bezpečnou instalaci, generování a spouštění. [25]

EAL3 je možno ještě dosáhnout bez podstatných změn základních existujících vývojářských praktik. Je aplikovatelná v případě, že se vyžaduje střední úroveň nezávisle ověřené bezpečnosti a je opřena o důkladné zkoumání TOE (ST, PP). Navíc oproti EAL2 se vyžaduje rozsáhlejší testování, kontroly vývojového prostředí a zajištění správy konfigurace. [25]

EAL4 stále umožňuje pohybovat se v rámci dobré komerční vývojářské praxe. Jakkoliv přísné jsou tyto praktiky, nevyžadují podstatné specializované znalosti, dovednosti a jiné zdroje. EAL4 je nejvyšší úrovní záruk, kterou lze dosáhnout (za rozumné náklady) zpětně pro již existující produkt. Poskytuje střední až vysokou úroveň záruky nezávisle ověřené bezpečnosti pro běžnou komoditu produktů a vyžaduje ze strany vývojáře nebo uživatelů připravenost k pokrytí dodatečných specifických nákladů spjatých s bezpečnostním inženýrstvím. Navíc oproti EAL3 se již vyžaduje také detailní návrh (low-level design) TOE, neformální model bezpečnostní politiky TOE a dodání určité podmnožiny implementace (např. část zdrojového kódu bezpečnostních funkcí). Nezávislá analýza zranitelností musí demonstrovat odolnost vůči průniku útočníků s nízkým potenciálem pro útok. Kontroly vývojového prostředí jsou doplněny modelem životního cyklu, stanovením nástrojů a automatizovanou správou konfigurace. [25]

EAL5 vyžaduje kromě přísného uplatnění dobré komerční vývojářské praxe aplikaci speciálních technik bezpečnostního inženýrství ve středním rozsahu. Dané TOE bude pravděpodobně již navrženo a vyvíjeno s cílem dosáhnout úrovně záruk EAL5. Nepředpokládá se nicméně velké zvýšení nákladů oproti EAL4. EAL5 je tak vhodná v případech, kdy se vyžaduje vysoká úroveň záruky nezávisle ověřené

bezpečnosti aniž by náklady na specializované techniky byly nerozumně vysoké. Navíc oproti EAL4 je vyžadováno dodání kompletní implementace TOE, formální model bezpečnostní politiky TOE, poloformální presentace funkčních specifikací, poloformální globální návrh (high-level design) a poloformální demonstrace korespondence. Nezávislá analýza zranitelností musí demonstrovat odolnost vůči průniku útočníků se středním potenciálem pro útok. Vyžaduje se také analýza skrytých kanálů a modularita návrhu. [25]

EAL6 vyžaduje aplikaci technik bezpečnostního inženýrství do přísného vývojového prostředí a je určena pro vývoj TOE sloužícího pro ochranu vysoce hodnotných aktiv proti význačným rizikům, kdy lze odůvodnit dodatečné náklady. Navíc oproti EAL5 se vyžaduje poloformální detailní návrh, rozsáhlejší testování, návrh TOE musí být modulární a zvrstvený, prezentace implementace strukturovaná. Nezávislá analýza zranitelností musí demonstrovat odolnost vůči průniku útočníků s vysokým potenciálem pro útok. Analýza skrytých kanálů musí být systematická. Vyšší nároky jsou kladeny na správu konfigurace a kontroly vývojového prostředí. [25]

EAL7 je použitelná pro vývoj produktů určených do extrémně rizikového prostředí a/nebo kde vysoká hodnota aktiv ospravedlňuje vyšší náklady. Praktické použití EAL7 je v současnosti omezeno na TOE a úzce vymezenou bezpečnostní funkčností, kde lze provést formální analýzu v požadované míře. Vyžaduje se plná formalizace, formální model bezpečnostní politiky, formální presentace funkčních specifikací and high-level návrhu, poloformální detailní návrh, formální a poloformální demonstrace korespondence. Testování se vyžaduje na úrovni bílé skříňky (white-box) a musí být dosaženo úplného nezávislého potvrzení výsledků všech předložených testů. Složitost návrhu musí být minimalizována. [25]

Operační systém Linux dosahuje úrovně EAL4. [26] [27]

7.4 Uživatelé a přihlášení

Linux je multiuživatelským systémem a model uživatelského oprávnění vychází z modelu DAC (viz. kapitola 7.2.1/s34). Přihlášení uživatele probíhá podobně jako u webových a jiných služeb. Tedy uživatelské heslo není v systému uloženo, ale je uložen pouze jeho HASH, získaný pomocí funkce SHA1 (záleží na nastavení sytému), když se pak uživatel přihlašuje do systému, je z jím zadaného hesla vygenerován HASH a porovnán s HASHem uloženým v sytému.

7.4.1 Uložení uživatelů a hesel

Uživatelské v OS Linux jsou definováni v souboru */etc/passwd* a jejich hesla jsou zabezpečena pomocí šifry SHA1 a uchována v souboru */etc/shadow*, který je přístupný pouze uživateli root.

/etc/passwd

Základní databází uživatelů v systému Linux je textový soubor */etc/passwd* (angl. password file), v němž jsou uvedena platná uživatelská jména a další k nim přidružené informace. Každému uživateli odpovídá v souboru jeden záznam - řádek, který je rozdělen na sedm polí, jejichž oddělovačem je dvojtečka. Význam jednotlivých položek je následující [5]:

1. Uživatelské jméno.
2. Pole pro heslo v zakódované podobě, které se nepoužívá a heslo se ukládá do */etc/shadow*.

3. Identifikační číslo uživatele.
4. Identifikační číslo pracovní skupiny (angl. group ID, zkráceně GID).
5. Skutečné jméno uživatele, případně popis účtu.
6. Domovský adresář.
7. Příkazový interpret (nebo program), který se spustí po přihlášení.

Každý uživatel systému má k souboru */etc/passwd* přístup (může jej číst). Může tedy například zjistit přihlašovací jména ostatních uživatelů [5].

/etc/shadow

Soubor */etc/shadow* (nebo také-li soubor stínových hesel) uchovává hesla uživatelů v zašifrované podobě pomocí definovaného šifrovacího algoritmu (nejčastěji MD5, DES nebo SHA1). Často se nepoužívá přímo šifra hesla, ale tzv. salt, což je ochrana proti tzv. rainbow attacku (česky duhový útok), což je útok na HASHovací funkce pomocí tabulek hesel (rainbow tables) a podobností v nich. Salt (česky sůl) pak rozpouští duhu a znemožňuje tak rainbow attack, díky přidání předdefinovaných stringů do hesla, které je pak zašifrováno. [51]

Můžou existovat tři druhy uživatelů podle možností přihlášení:

Uživatel s heslem běžný uživatel s nastaveným heslem. Jeho nastavení vypadá asi takto:
/etc/passwd

```
lfs:x:1002:1000::/home/lfs:/bin/bash
```

/etc/shadow

```
lfs:$1$MhyAxheU$Nq85ITaTmL8yJqgHXoOEK0:15111:0:99999:7:::
```

Uživatel bez hesla Běžný uživatel jehož přihlášení nevyžaduje heslo, **POZOR! je rozdíl jestli je uživatel bez hesla nebo se při startu systému přihlašuje uživatel automaticky prostřednictvím správce sezení jako je GDM nebo KDM**. Nastavení uživatele bez hesla se liší pouze v souboru /etc/shadow a heslo je nahrazeno prázdným stringem:
/etc/shadow

```
lfs:$1$VNMbpxGH$sew7cnwH9ixU.x27UbFNn.:15111:0:99999:7:::
```

POZOR!!! takový přístup se nepoužívá! Uživatelé bez hesla se používají pouze na LiveCD!

Uživatel bez přihlášení uživatel bez přihlášení se používá pro služby (v Linuxu daemony, česky někdy též démony), je to speciální druh uživatele, za kterého se nelze přihlásit a používá se aby byla práva služeb oddělena od uživatelů a neběžela pod oprávnění uživatele root. Jejich nastavení se liší od normálního uživatele pouze v tom, že v souboru */etc/passwd* je defaultní shell jako */bin/bash* nahrazen */bin/nologin* nebo */bin/false*
/etc/shadow

```
dbus:x:81:81:System message bus:/:/bin/false
```

a heslo v */etc/shadow* je nahrazeno vykřičníkem

```
dbus:!:15084:0:99999:7:::
```

7.4.2 Přihlášení uživatele

Pro přihlášení uživatele se používá několik způsobů, ať už grafických textových nebo vzdálených.

Základní možnosti přihlášení uživatele:

Computer terminal (tty) Standardní přihlášení do systému, uživatel zadá uživatelské jméno, heslo a je mu zpřístupněn systém prostřednictvím console.

Desktop manager slouží jako přihlašovací obrazovka do většiny grafických prostředí a většinou poskytuje kromě přihlášení, také služby jako je vypnutí, restart nebo uspání bez nutnosti zadání hesla uživatele root. Existují **grafické desktopové managery** jako KDM, GDM, SLiM, XDM, EDM, LightDM a jiné. Velmi často se ke grafickému prostředí váže i jeho vlastní desktop manager, který ovšem dokáže spouštět i ostatní prostředí. **Textových desktopových managerů** existuje velmi málo a většinou se jedná pouze o nástroje, které umožní po textovém přihlášení spustit Xserver se zvoleným grafickým prostředím. wmctrl, CDM.

Vzdálené přihlášení vzdálených přihlášení existuje mnoho druhů, protože spousta služeb, které umožňují vzdálený přístup implementuje vlastní způsob přihlášení. Např. některé grafické desktopové managery umožňují vzdálený přístup přes protokol XDMCP. Nejčastěji se však pro vzdálené přihlášení používá SSH (viz. kapitola 7.5.3/s44)

7.5 Internetové zabezpečení

7.5.1 Firewall v Linuxu

Firewall v Linuxu je tvořen projektem **Netfilter**, který pracuje na úrovni jádra a umožňuje filtrovat pakety na základě mnoha kritérií. Základním nástrojem pro nastavení paketového filtru je známý řádkový nástroj **iptables**. [33]

IPtables jsou ale nástrojem velice složitým a proto vzniklo několik projektů, které se snaží nastavení Firewallu usnadnit. Nástrojem určeným pro běžné uživatele je např. grafický program **Firestarter**. Přístup Firestarteru je však vhodný spíše pro osobní počítače, pro větší síť a nastavení více firewallů, je vhodným řešením nástroj, ve kterém si uživatel vytvoří nastavení v grafickém nebo textovém rozhraní a výslednou konfiguraci firewallu pak přenese na server. Takovým nástrojem je např. **FirewallBuilder**, **FireHOL** oblíbená sada skriptů pro konfiguraci Netfilteru **Shorewall**.

Jak už je v Linuxu dobrým zvykem vše je často přehledně a použitelně zabaleno v distribucích. V prostředí firewallů existuje velké množství distribucí, které nenabízejí většinou v základu jen firewall, ale jsou určiny i pro routery a jiná síťová zařízení (PROXY server, gateway, VPN, IPSEC, Anti-virus, www server,…). Asi nejpoužívanější distribucí je **IPCop**, což je derivát distribuce **Smoothwall**. Oblíbenou distribucí pro firewally je také router distribuce **OpenWRT** a několik projektů na ní založených např. **FreeWRT** nebo **DD-WRT**.

7.5.2 Antivirus

Antivirus a viry obecně jsou velmi diskutovaným tématem v Linuxovém prostředí. V dnešní době je nebezpečí virů na Linuxu stále mizivé a vše naznačuje, že to tak ještě nějakou dobu potrvá. Hlavním argumentem proč na Linuxu neexistují viry je často jeho malá rozšířenost, což není tak úplně pravda, protože většina průzkumů rozšířenosti OS se zaměřuje na desktopové počítače, kde dle statistik Linux drží 1% [34], ale co se týče především www serverů dosahuje Linuxový webový server Apache podílu přes 70% [35]. Linux je před viry a malwarem chráněn především díky jeho balíčkovacím systémům, protože software není jako u windows instalován z různých pochybných webů a jiných zdrojů, ale přímo od tvůrců distribuce a jejich repozitářů. Dalším stupněm ochrany

jsou pak oddělené uživatelské účty a oddělený účet administrátora, které zabraňují šíření viru v systému a v jeho dalším napadení. [37]

Samozřejmě pro Linux existuje množství antivirových programů, které spousta neznalých uživatelů považuje za důkaz virů v Linuxu, což není pravda, protože antivirové programy v Linuxu slouží především k ochraně Windowsových stanic. Většina serverových řešení je totiž postavena právě na Linuxu a je třeba chránit klienty, kteří fungují na Windows. Nejznámější antivirovým programem pro Linux je zřejmě **ClamAV**, jedná se o OpenSource program, který slouží především ke scanování e-mailové komunikace a ochraně windowsových stanic [36]. Samozřejmě existuje i množství komerčních antivirových programů od firem jako ESETs nebo AVG a je jen na uživateli, které řešení použije.

7.5.3 SSH

Secure Shell neboli ssh je nástroj pro bezpečné připojení v nedůvěryhodné síti. Poskytuje šifrované terminálové spojení s bezpečným systémem autentizace, jak na straně serveru tak straně klienta, pomocí veřejných kryptografických klíčů.
Hlavní výhody:

- Množství autentizačních metod
  - **Heslem** základní zabezpečení, které závisí na znalosti hesla.
  - **Veřejným klíčem** ověřená prostřednictvím kryptografických klíčů. Bezpečnější než přihlašování heslem.
  - **Rhosts** podle důvěryhodných hostitelů, povoluje připojení na základě adresy hostitelského počítače. Nepovažuje se za bezpečné.
  - **RhostsRSA** varianta předcházejícího, hostitelé se navíc musí autorizovat kryptografickým klíčem RSA.
  - SSH podporuje i další způsoby jako například Kerberos, SecurID, S/Key, PAM a podobně.
- tunelování libovolné TCP spojení přes SSH, chrání obvykle nešifrované protokoly, jako je IMAP a umožňuje bezpečný průchod přes firewally
- automatické přeposílání spojení X windows
- secure file transfers (bezpečné přenášení souborů)

SSH znamená kromě možnosti snadného přístupu uživatele do vzdáleného počítače i cestu pro nějakého útočníka. Proto je při používání SSH doporučeno několik bezpečnostních postupů, které snižují možnost napadení počítače nejčastěji pomocí slovníkových útoků.

Používat veřejných klíčů použití veřejných klíčů zabraňuje slovníkovému útoku. např. GNU Privacy Guard (GPG)

Zakázat přihlášení některých uživatelů především se jedná o účet uživatele root, který je v každém Linuxu. Většina slovníkových útoků se zaměřuje právě na tohoto uživatele. Další časté pokusy jsou zaměřeny např. na uživatele user, admin, test,…

fail2ban je program, který vyhodnocuje log s pokusy o přihlášení prostřednictvím SSH a na základě neúspěšných přihlášení z unikátní IP adresy, zakazuje tuto IP adresu na přednastavenou dobu.

7.6 Hypotetické bezpečnostní hrozby v Linuxu

Virové hrozby byli probrány (viz. kapitola 7.5.2/s43) v souvislosti s viry a malwarem se, ale v Linuxu mluví o hrozbě jménem **sudo**, sudo je nástroj přes který si uživatel nebo aplikace může požádat o vyšší oprávnění než jsou jí určeny, k samotnému přidělení oprávnění je třeba aby uživatel zadal heslo pro uživatele root, což zdánlivě celou situaci řeší, ale zde vstupuje nejzranitelnější část celého systému a tou je uživatel. Neznalý uživatel zadá bez rozmyslu svoje heslo a aplikace má plný přístup nad systémem. Samozřejmě jedná se o faktor, který vývojář nevyřeší, ale hrozba číhá i někde jinde, celkem jednoduše jde vytvořit alias[4)] na program sudo a tak spustit při zadání příkazu sudo nebezpečný kód pod oprávněním uživatele root. např.

```
alias sudo='sudo rm -rf /'
```

Při takovém aliasu by si uživatel smazal disk. Samozřejmě takový kód je jen na ukázku, ale demonstruje možnosti tohoto "útoku", daleko pravděpodobnější by bylo spuštění útočníkova skriptu a následné přepsání konfiguračních souborů, tak aby získal útočník kontrolu nad počítačem. Tomuto se dá zabránit aplikování MAC, ale využití podobného rozšíření zabezpečení, není v Linuxu vůbec samozřejmostí.

Lidský faktor přispívá ke zranitelnosti jakéhokoliv systému, ukázkou může být chyba vývojáře debianu, která způsobila, že autentizační mechanizmus u ssh generoval předvídatelné textové řetězce, které mohl zneužít útočník pro napadení systému. Tato chyba se navíc vyskytovala v debianu po dlouhou dobu než se na ni přišlo. [39]

Další hrozbou jsou pak chyby v kódu jádra. Naštěstí je, díky otevřenému kódu, kolem Linuxu obrovská skupina vývojářů, která chyby téměř okamžitě záplatuje a eliminuje tak zero-day útoky.

Repozitáře s falešným nebo upraveným SW jsou další z reálných hrozeb v Linuxu, avšak uživatel je opět ten, který si přidá takovýto repozitář. Této možnosti útoku zabraňuje podepisování balíčků a repozitářů. [40]

Existuje ještě mnoho hrozeb, ale většinu neovlivňuje Linux samotný, ale různé aplikace třetích stran a chyby v nich obsažené.

8 SYSTÉM PRO SPRÁVU VERZÍ GIT

Projekt git byl založen v roce 2005 Linusem Torvaldsem, původně pro vývoj jádra Linuxu, a jsou do něj vloženy zkušenosti, které Linus získal při vedení rozsáhlého projektu Linuxového jádra. [53]

Git je zcela zdarma, open source, **distribuovaný systém pro správu verzí** navržený zvládnout vše od malých až po velmi rozsáhlé projekty s rychlostí a účinností. Každý klon Git je full-fledged (česky plně rozvinuté) úložiště s kompletní historií a veškerými schopnostmi sledování revizí a nezávislostí na přístupu k síti nebo centrálnímu serveru. Větvení (branching) a slučování (merging) jsou rychlé a snadné. [54] Velmi důležité je pak slovo distribuovaný, protože git neukládá každou změnu na centrální úložiště, ale každý klon repozitáře obsahuje i kompletní databázi změn projektu.

8.1 Použití git v bakalářské práci

Ke psaní této bakalářské práce byl zvolen sázecí systém LaTeX, jednak pro jeho velmi kvalitní a typograficky správný výstup, ale i protože je taková práce psána formou zdrojového kódu. Díky uložení zdrojového kódu do plain/text souboru je velmi snadné a výhodné použít systému pro správu verzí. Systém git byl také použit pro správu verzí skriptů pro praktickou část „LFS by BASH scripts“, které jsou uloženy také v textových souborech.

[4)] alias je, jak název napovídá, taková přezdívka, uživatel si může pod řetězec uložit složitější příkaz, např. alias ls='ls -hF –color=always'

8.2 Základní příkazy a sociální web github

Ke správě jednoduchých a malých projektů jako je např. tato bakalářská práce stačí umět opravdu jenom pár příkazů. A díky webu github.com se stává správa malého projektu opravdu jednoduchou záležitostí.

8.2.1 Nastavení gitu

Základní nastavení gitu spočívá pouze v nastavení Jména, Příjmení a emailu. Pokud používáme git pouze lokálně a nenahráváme data někam na server, bude nám to stačit a můžeme již vytvořit repozitář.

Nastavení jména, příjmení a emailu. [55]

```
git config --global user.name "Jméno Příjmení"
git config --global user.email "VášEmail@seznam.com"
```

Pokud nahráváme data na server je třeba autentizace. K účelu autentizace se často používá ssh klíč. Ten je možné vygenerovat následujícím způsobem [55]

```
ssh-keygen -t rsa -C "VášEmail@seznam.com"
```

pak je třeba zadat dvakrát heslo pro otevírání klíče a klíč je vygenerován.

Vygenerovaný klíč je k nalezení v domovském adresáři uživatele ve složce *.ssh*, která obsahuje soubory *id_rsa* a *id_rsa.pub*. id_rsa je váš privátní klíč a id_rsa.pub je klíč veřejný. Pro použití githubu je třeba veřejný klíč nahrát do nastavení na webu *"Account Settings" > Click "SSH Public Keys" > Click "Add another public key"*. Také je potřebné nahrát do nastavení gitu bezpečnostní token githubu. Bezpečnostní token lze nalézt na webu *"Account Settings" > Click "Account Admin."* položku API token, která obsahuje jakýsi HASH, je třeba vložit do nastavení git na počítači. [55]

```
git config --global github.user uzivatelskeJmeno
git config --global github.token 0123456789yourf0123456789token
```

8.2.2 Vytvoření repozitáře a první commit

Vytvoření repozitáře je velmi jednoduché a skládá se ze dvou kroků: Vytvoření složky a inicializace repozitáře.

```
mkdir projekt
cd projekt
git init
```

Změny v repozitáři jsou reprezentovány commity. Commit je vytvořen přidáním jednoho nebo více souborů a přidáním popisku. Jako první by měl projekt obsahovat soubor README.

```
touch README
git add README
git commit -m 'přidáno README'
```

Pokud chceme použít github, je třeba vytvořit repozitář na webu github. Po vyplnění názvu a popisu repozitáře, github vypíše kroky nutné k vytvoření repozitáře. Vypíše tedy, že je třeba vytvořit složku, inicializovat git, přidat první commit (nejčastěji soubor README), přidat adresu repozitáře na serveru github a odeslání změn v repozitáři. [55]

```
git remote add origin git@github.com:uzivatelskeJmeno/projekt.git
git push origin master
```

8.2.3 Správa repozitáře

Každodenní správa repozitáře jednoduchého projektu je většinou o přidávání commitů a odesílání změn. Takže si většinou lze vystačit s příkazy *add, rm, diff, commit, push a pull*. Ale git obsahuje také spoustu jiných užitečných funkcí, které usnadní správu i malého projektu.

Commit není pouze přidávání souborů, ale i mazání a změny v souborech.

```
git rm soubor #smaže soubor
git add #přidá soubor, pokud soubor změníme je třeba ho také přidat
git diff #ukáže změny v repozitáři
git commit #vytvoří commit
git push #odešle změny
git pull #stáhne změny ze serveru
```

Jednou z funkcí je **tag**. Tag vytvoří značku za určeným commitem ke které se lze jednoduše vrátit. Je tedy užitečný pokud chce vytvořit např. release programu. commity a jejich kódy si lze prohlédnout příkazem *git log*

```
git tag v1.0 <kód commitu>
```

K tagu se lze jednoduše vrátit příkazem *checkout*. (příkaz *chekcout* také mění větve)

```
git checkout v1.0
```

Další funkcí, která je užitečná je branch (větev). Branch můžeme např. použít pokud se chystáme udělat změnu, o které nevíme jistě, zda bude fungovat. Často je branch také k oddělení prací uživatelů nebo částí projektu na kterých se pracuje.
Branch lze vytvořit příkazem

```
git branch <název větve>
```

Repozitář je automaticky přepnut na vytvořenou větev a změny už se neprovádí v hlavní větvi master, ale v nové větvi. Vytváření commitů a odesílání změn je pak normální. Nastane ale čas, kdy je třeba části větve nebo větev celou začlenit do větve hlavní. K tomuto je několik příkazů: první z nich je *rebase*, který stáhne změny z hlavní větve a aplikuje je před změny ve vytvořené větvi. Pokud v projektu nevznikají změny souběžně v hlavní větvi i větvi nové, není rebase třeba. Změny aplikujeme pomocí změny větve zpět na master příkazem *checkout* a odeslání změn z vytvořené větve pomocí *pull*. (takto lze aplikovat změny mezi libovolnými větvemi)

```
git checkout -f master
git pull . <název větve>
```

Samozřejmě může nastat čas, kdy větev již není třeba a chceme ji spojit s hlavní větví nebo smazat.

```
git merge <název větve> #spojí aktuální větev se zadanou
git branch -D <název větve> #smaže větev
```

Pokud souběžně vznikají změny ve větvi hlavní i nové větvi a chceme provést příkazy jako *rebase, pull . <větev>* nebo *merge* mohou nastat konflikty ve změnách, které je třeba vyřešit (git některé konflikty řeší sám, ale nedokáže všechno). V takovém případě přepne git aktuální větev do speciálního režimu a předá uživateli informace k vyřešení těchto konfliktů. [56]

II. PROJEKTOVÁ ČÁST

9 LFS BY BASH SCRIPTS

Účelem LFS by bash scripts je sestavení distribuce LFS za pomocí jednoduchých a přehledných skriptů napsaných v interpretovaném jazyce BASH (viz. kapitola 4/s17). Jejich struktura je velice jednoduchá a přehledná tak, aby ji kdokoliv mohl libovolně upravovat bez velkého zkoumání kódu.

9.1 Struktura a použití

V git repozitáři LFS—by-bash-scripts je k nalezení několik souborů a složek.

9.1.1 Adesářová struktura

chap5 Obsahuje skripty pro sestavení toolchainu dle kapitoly 5 v oficiálním návodu.

chap6 Obsahuje skripty pro sestavení systémových programů v prostředí chroot dle kapitoly 6 v oficiálním návodu.

chap7 Obsahuje skripty pro nastavení bootscripts.

chap8 Obsahuje skripty pro poslední fázi sestavení.

ChrootIN Obsahuje skripty pro chroot do systému v různých fázích sestavení

FAIL_Logs Složka, která slouží pouze pro ukládání chybových záznamů pokud některá část kompilace selže.

include Adresář s funkcemi, jazykovou složkou a verzemi programů

lfsuser Pro usnadnění práce tento adresář obsahuje soubory potřebné pro uživatelský profil LFS.

chroot_makeall Hlavní spouštěcí skript pro sestavení kapitoly 6 v chroot.

linker.1.test/linker.1.test64 Dva soubory, které slouží pro test funkčnosti při oddělení toolchainu.

makeall Hlavní spouštěcí skript, který spustí vše potřebné až do konce kapitoly 5.

makeall_rest Hlavní spouštěcí skript pro sestavení kapitoly 7 a 8.

onlyunpackscript Skript pro stažení a rozbalení balíčků.

README.markdown README soubor ve značkovacím jazyce markdown

sources Pozůstatek složky pro stahování zdrojových kódů před instalací. Už řešeno přímo před kompilací jednotlivě.

version-check.sh Oficiální skript LFS pro test potřebných nástrojů v systému

Velmi důležitou je pak složka include, která obsahuje verze programů a funkce využívané při kompilaci

LFS—by-bash-scripts/include

chroot_prepare Skript, který se stará o kontrolu potřebných složek a symbolických odkazů v prostředí chroot.

functions skript se všemi funkcemi.

prepare Skript, který se stará o kontrolu existence potřebných složek a symbolických odkazů.

ver Skript s poli, které obsahují informace o balíčcích

l10n-EN Už téměř nepoužívaný skript, který měl sloužit pro lokalizaci do jiných jazyků, byl zavrhnut, protože znamenal velkém množství proměnných

9.1.2 Důležité proměnné

LFS samotná obsahuje několik proměnných, které jsou důležité pro sestavení toolchainu a LFS by bash scripts obsahuje další proměnné důležité pro funkčnost skriptů.

jsou to především:

TOOLS_STRIP=0 Proměnná nastavuje, zda se mají odstranit některé nepotřebné soubory v toolchainu, vhodné pro malý diskový oddíl.

WAIT=1 Proměnná nastavuje, zda se má v každém kroku čekat na uživatelskou interakci. Doporučena hodnota 1 (čekat)

MAKEFLAGS='-j 3' Tato proměnná nastavuje počet jader procesoru.

ROOT=/mnt/LFS Cesta, na které je připojen disk

SOURCES_DIR=$ROOT/sources Cesta, na které jsou umístěny stažené balíčky

BUILD_DIR=$ROOT/build Cesta do adresáře, kde probíhá kompilace

TOOLS_DIR=$ROOT/tools Cesta do adresáře, kde je instalován toolchain

SCRIPTS_DIR=$ROOT/scripts Cesta do adresáře, kde jsou umístěny skripty

on64=0 Pokud je systém sestavován na stroji s 64bitovým procesorem, ale na 32bitovém sytému je třeba nastavit na 1.

Každý skript ke kompilaci programu navíc obsahuje interní proměnné, do kterých se ukládají názvy programů jejich adresy ke stažení a jiné informace z polí, uložených v souboru *includes/ver*. Tyto proměnné se pak většinou předávají jako parametry funkcím definovaným v souboru *includes/funtions*.

malá ukázka ze skriptu chap5/glibc:

NAME= ${glibc[0]} do proměnné se ukládá název programu z první položky pole (glibc)

PROGRAM= ${glibc[3]} do proměnné se ukládá název programu i s jeho verzí, slouží především k určování adresy po rozbalení programu. (glibc-2.13)

FILE= ${glibc[2]} do proměnné se ukládá název archivu s programem (glibc-2.13.tar.bz2)

EXT= ${glibc[4]} do proměnné se ukládá druh archivu, který rozhoduje ve funkci unpack() (viz. kapitola 9.1.3/s51), o tom jak se má rozbalit. (glibc-2.13.tar.bz2)

DLINK= ${glibc[1]} do proměnné se ukládá odkaz na stažení archivu využívaný funkcí download() (viz. kapitola 9.1.3/s53) (http://… ftp://…)

MD5= ${glibc[5]} do proměnné se ukládá kontrolní součet staženého souboru pro test správnosti stažení, využívaný funkcí check (viz. kapitola 9.1.3/s53) (38808215a7…)

pDLINK/pMD5/pPatch pro patche jsou proměnné podobné, jen pole obsahuje pouze 3 položky a proměnná pole začíná písmenem "p". položka[0] obsahuje link pro stažení, položka[1] obsahuje název souboru s patchem, položka[2] obsahuje kontrolní součet.

9.1.3 Functions

Skript functions slouží pro automatizování stále se opakujících procedur pomocí BASH funkcí. Každá funkce obsahuje jednoduchý komentář s popisem funkcí a jejich parametrů v angličtině. Pro ukázku několik funkcí z tohoto skriptu.

Pro barevný výstup jsou definovány barvy, které je třeba pro použití ve funkcích nastavovat přes program **tput**

```
DEF="$(tput sgr0)"
BOLD="$(tput bold)"
B="${BOLD}$(tput setaf 4)"
G="${BOLD}$(tput setaf 2)"
R="${BOLD}$(tput setaf 1)"
Y="${BOLD}$(tput setaf 3)"
C="${BOLD}$(tput setaf 6)"
M="${BOLD}$(tput setaf 5)"
```

unpack() je jednoduchá funkce, která pouze na základě předaného parametru z proměnné $EXT volá funkce pro rozbalení.

```
unpack(){
if [ $1 == bzip ]; then
      unpackBzip
  elif [ $1 == gzip ]; then
      unpackGzip
  else
    echo "$R somethink went wrong with archive unpack $DEF";
fi
}
export -f unpack
```

unpackBzip() a unpackGzip() jsou prakticky stejné funkce, které rozbalují stažené archivy, jejich rozdíl je pouze v parametrech programu tar.

```
unpackGzip (){
  echo "$Y Starting build script $PROGRAM $DEF";
  if [ -d $BUILD_DIR/$PROGRAM ]; then
   echo "$C Directory with source already exist $G [REMOVING] $DEF"
    echo "$G rm -rf $BUILD_DIR/$PROGRAM $DEF"
    rm -rf $BUILD_DIR/$PROGRAM
    echo "$C Source directory $G [UNPACKING] $DEF"
    echo "$G tar xzf $SOURCES_DIR/$FILE -C $BUILD_DIR/ $DEF"
    tar xzf $SOURCES_DIR/$FILE -C $BUILD_DIR/
  else
    echo "$C Source directory $G [UNPACKING] $DEF"
    echo "$G tar xzf $SOURCES_DIR/$FILE -C $BUILD_DIR/ $DEF"
    tar xzf $SOURCES_DIR/$FILE -C $BUILD_DIR/
  fi
}
export -f unpackGzip
```

SepBuild() vytváří složku pro oddělený build. Některé programy není dobré kompilovat ve složce se zdrojovými kódy a proto jsou kompilovány v separované složce.

```
SepBuild (){
        echo "$Y Preparing for separate build $PROGRAM $DEF";
        if [ -d $BUILD_DIR/$PROGRAM-BUILD ]; then
        echo "$C Build directory already exist $G [REMOVING] $DEF"
          echo "$G rm -rf $BUILD_DIR/$PROGRAM-BUILD $DEF"
          rm -rf $BUILD_DIR/$PROGRAM-BUILD
        else
          echo "$M.$DEF"
        fi
        echo "$C Making directory for separate build $DEF"
        echo "$G mkdir $BUILD_DIR/$PROGRAM-BUILD $DEF"
        mkdir $BUILD_DIR/$PROGRAM-BUILD
}
export -f SepBuild
```

testSimLink() je funkcí pro testování existence symbolického odkazu. První parametr $1 je testovaný odkaz a pokud neexistuje, je vytvořen symbolický odkaz parametru $2 na adresu v parametru $3

```
testSimLink (){
        if [ -L $1 ]; then
          echo "$C Symbolic link $1 exist $DEF"
        else
          echo "$C Symbolic link $G $1 $C not exist $DEF"
          echo "Making symbolic link $G $1 $DEF"
          echo "$G ln -s $2 $3 $DEF"
          ln -s $2 $3
        fi
}
export -f testSimLink
```

Funkce addPatch() využívá programu **patch**, pro aplikování patche. A jejím jediným parametrem je název souboru patche, který je většinou v proměnné $pPatch.

```
addPatch (){
        echo "$C Adding patch $DEF"
        echo "$G patch -Np1 -i $SOURCES_DIR/$1 $DEF"
        patch -Np1 -i $SOURCES_DIR/$1
}
export -f addPatch
```

Funkce compareFiles() porovnává dva soubory pomocí programu **cmp**. Je vytvořena pro účely testování správnosti oddělení toolchainu a pokud se soubory neshodují, ukončuje skripty.

```
compareFiles (){
if cmp $1 $2 &> /dev/null
        then
          echo "$C Files are matching [OK] $DEF"
        else
          echo "$R Linker test [FAIL] $DEF"
          exit 1
    fi
}
export -f compareFiles
```

Funkce waitUser() na základě proměnné $WAIT pozastavuje vykonávání skriptu, dokud uživatel nezmáčkne klávesu enter. Podobnou funkcí je pak fuknce waitUserAl(), která pozastaví skripty vždy.

```
waitUser (){
  if [ $WAIT == 1 ]; then
    echo "for continue pres enter"
    read > /dev/null
  fi
}
export -f waitUser
```

Již zmíněná funkce download() stahuje z internetu všechny potřebné soubory a využívá k tomu programu wget.

```
download (){
 wget -c ${1} -P ${SOURCES_DIR}/
}
export -f download
```

Funkce check() uloží do proměnné $md5test kontrolní součet souboru, předaného v parametru $1 a porovná jej s kontrolním součtem předaným v parametru $2.

```
check(){
  echo "$C Checking md5 checksum for file ${1} $DEF"
  md5test=$(md5sum $SOURCES_DIR/${1} | awk '{ print $1 }')
  if [ ${2} == ${md5test} ]; then
    echo "$G [OK] $DEF";
  else
    echo "$R [FAIL] $DEF";
  fi
}
```

Functions obsahuje také funkce pro kompilaci, ale předávání parametrů pomocí $ se ukázalo jako nešťastné řešení, způsobující množství chyb při kompilaci.

9.2 Návod pro sestavení

9.2.1 Předpříprava

Sestavení LFS je velmi komplexní a složitou záležitostí. V žádném případě by neměl používat automatické skripty ten, kdo předtím úspěšně nesestavil LFS alespoň dvakrát. Pro člověka, který nikdy nesestavoval LFS je v balíčku skriptů skript **onlyunpackscript**, který zajišťuje stáhnutí potřebných balíčků a jejich rozbalení více (viz. kapitola 9.2.4/s58).

Co by měl uživatel udělat před sestavení LFS pomocí automatických skriptů

- několikrát úspěšně sestavit LFS
- přečíst znovu alespoň první 4 kapitoly (je dobré číst knihu za běhu skriptů)
- zkontrolovat potřebné balíčky. Ke kontrole slouží skript version-check.sh, který zkontroluje verze některých balíčků a potřebné symbolické odkazy
- být připraven řešit spoustu problémů. Při sestavování LFS může nastat v kterékoliv části problém, který je třeba vyřešit k dalšímu pokračování. Kam se obrátit z dotazem je popsáno (viz. kapitola 9.2.5/s59)

Uživatel by měl mít také dostatečné znalosti jazyka BASH, aby mohl v případě potřeby upravit skripty.

9.2.2 Sestavení

Sestavení je díky skriptům vcelku jednoduchý proces. Podle návodu v LFS uživatel vytvoří prázdný diskový oddíl nejjednodušeji zřejmě grafickým nástrojem jako gparted. Vytvoří složku do které se oddíl připojí a vytvoří proměnnou, která definuje cestu k této složce. Pozn. /dev/sda7 musí být nahrazeno správným oddílem, k dohledání tohoto oddílu, pokud jsme například měli již oddíl vytvořen a nepamatujeme si jeho adresu, může pomoci program blkid. (všechny tyto kroky musí být provedeny pod účtem uživatele root),

```
# mkdir /mnt/LFS
# export LFS=/mnt/LFS
# mount /dev/sda7 $LFS
```

poté je potřebné vytvořit složku pro toolchain a symbolický odkaz na adresu /tools

```
# mkdir -v $LFS/tools
# ln -sv $LFS/tools /
```

Dalším krokem je vytvoření uživatele a skupiny lfs.

```
# groupadd lfs
# useradd -s /bin/bash -g lfs -m -k /dev/null lfs
# passwd lfs #nastavení hesla
```

Nyní je nutné stáhnout a překopírovat skripty LFS by bash scripts. Nejjednodušším způsobem je stáhnutí skriptů pomocí verzovacího systému git. Repozitář může obsahovat změny, které by mohly udělat návod nefunkčním, proto je třeba změnit branch na tag BCv6.8, na který návod sedí.

```
cd ~/
git clone git://github.com/zajca/LFS---by-bash-scripts.git
git checkout BCv6.8
# mkdir $LFS/{scripts,build,sources}
# cp -R ~/LFS---by-bash-scripts/* $LFS/scripts/
```

Dále překopírovat soubory uživatele lfs do jeho domovského adresáře.

```
# cp $LFS/scripts/lfsuser/{.bash_profile,.bashrc} /home/lfs/
# chown -R LFS /home/LFS
```

Nyní je třeba přenastavit práva tak aby uživatel lfs mohl zapisovat do složky /mnt/LFS

```
# chown -R lfs:lfs $LFS
```

Po těchto krocích je už třeba jen přihlásit se jako uživatel lfs.

```
su - lfs
cd $LFS
```

Protože pro stažení balíčků se používá program wget, který v toolchainu, respektive při sestavování kapitoly 6, nebude dostupný, je dobré stáhnou všechny balíčky skriptem onlyunpackscript (viz. kapitola 9.2.4/s58).

Nyní začne sestavení toolchainu, proto spustíme skript makeall.

```
sh ./scripts/makeall
```

Skript makeall ověří správnost nastavení a vytvoří toolchain, který je potřebný pro další sestavení systému. Jakmile je toolchain sestaven je třeba změnit práva pro adresář /mnt/LFS/tools

```
chown -R root:root $LFS/tools
```

Nyní je třeba spustit skript start6, který vytvoří potřebné složky a virtuální souborové systému. Skript také provede přihlášení do systému prostřednictvím prostředí chroot.

```
# sh ./scripts/chap6/start6
```

Nyní je dobré nastavit opět potřebné proměnné v souboru scriptschroot_prepare a spustit skript chroot_makeall.

```
./scripts/chroot_makeall
```

Skript je přerušen pro opětovné přihlášení a je potřeba ho spustit znovu.
Při kompilaci balíčku **glibc** je skript pozastaven aby bylo možné zkontrolovat možné chyby při kompilaci, **tato část je velmi důležitá pro funkčnost systému.** Podobné chybové hlášení je v pořádku. (chybový log je /build/glibc-2.13/glibc-check-log) Pokud se objeví více chyb je třeba je zkonzultovat (viz. kapitola 9.2.5/s59). Ale i více chyb neznamená, že bude systém nefunkční, chyby se často odvíjejí od HW.

```
make[2]: [/build/glibc-BUILD/posix/annexc.out] Error 1 (ignored)
make[2]: *** [/build/glibc-BUILD/rt/tst-cpuclock2.out] Error 1
make[1]: *** [rt/tests] Error 2
make: *** [check] Error 2
```

Po balíčku glibc proběhne oddělení systému od toolchainu, toto je **extrémně důležitá část**, skript je několikrát přerušen, je vždy vypsána hodnota v systému a hodnota, která by měla být vypsána, hodnoty se liší v závislosti na tom jestli se jedná o systém 64bitový nebo 32bitový. **Ve výpisech by se neměla objevit žádná chyba!**

Ihned po balíčku glibc je spuštěn skript chap6/locales, který kromě testu jazyků nastavuje i časovou zónu. Je třeba otevřít nové terminálové okno, přihlásit se do systému pomocí skriptu:

```
./scripts/chrootIN/chrootChap6
```

A v novém terminálovém okně spustit **tzselect**. Tzselect se zeptá na dvě otázky a vygeneruje časovou zónu, např. Europe/Prague, kterou je třeba zadat do předchozího terminálu a potvrdit klávesou enter.

Po testu u balíčku **binutils** by měl být zkontrolován jeho log, skript k tomu uživatele sám vyzve. Stejná situace nastane později u balíčku **mpfr**

Další přerušení kvůli důležitému testu nastane při balíčku **qmp**, skript je opět zastaven a na obrazovku jsou vypsány výsledky testů. Všech 162 testů by mělo být úspěšných, pokud ne je třeba tyto chyby zkonzultovat (viz. kapitola 9.2.5/s59).

Balíček **gcc** sebou nese velké množství testů, první test trvá opravdu dlouhou dobu a jeho výsledkem je množství hlášení. Tato hlášení je také možné konzultovat (viz. kapitola 9.2.5/s59), ale ani vývojáři LFS většinou neví, co který test přesně dělá a proč selhal. Ve většině případů je možné tato selhání ignorovat a pokračovat dále. O to důležitější je série testů po instalaci balíčku. **Tyto testy kontrolují správnost právě nainstalovaného překladače, takže je naprosto nezbytné je zkontrolovat!!!**. Probíhají úplně stejně jako testy u oddělení systému od toolchainu.

Po nainstalování balíčku **bash** je skript přerušen znovu, protože je uživatel přihlášen s právě nainstalovaným interpretem. Proto je třeba znovu spustit skript /scripts/chroot_makeall.

Poslední zastavení v kapitole 6 je pak u balíčku **shadow**, při přerušení skriptu je třeba otevřít nový terminál, přihlásit se prostřednictvím skriptu:

```
./scripts/chrootIN/chrootChap6
```

a v novém terminálovém okně nastavit heslo uživatele root příkazem:

```
passwd root
```

Po sestavení několik dalších balíčků je skript *chroot_makeall* ukončen a je možnost, stejně jako na konci kapitoly 5, zmenšit velikost systému smazání debugovacích symbolů. K tomuto je třeba se přihlásit do systému a spustit skript pro smazání debugovacích symbolů.

```
./scripts/chrootIN/chrootForStrip
./scripts/chap6/StripDebug
```

!!!NÁSLEDUJÍCÍ FÁZE OBSAHUJE NĚKTERÁ DŮLEŽITÁ NASTAVENÍ A JE TŘEBA, ABY JIM UŽIVATEL DOBŘE ROZUMĚL, PROTO JE DOPORUČENO NEPOUŽÍT SKRIPTY, ALE POKRAČOVAT DÁLE PODLE KNIHY!!!

Nyní je třeba se naposled přihlásit do systému a spustit skript na sestavení posledních dvou kapitol.

```
./scripts/chrootIN/chrootAfterChap6
./scripts/makeall_rest
```

Skript makeall_rest provede uživatele kapitolami 7 a 8. Několikrát skript vyzve uživatele k zadání potřebných údajů.

- Natavení locale, je třeba aby si uživatel vybral ze seznamu jazyk a kódování, např. en_US.utf8.
- Nastavení hostname, zde je třeba zvolit název pro systém (počítač).
- Nastavení network interface, zde je třeba aby uživatel vybral z výpisu identifikátor své síťové karty a vepsal jej do terminálu.
- Po skriptu s nastavení sítě je třeba nastavit síť. V souboru */etc/sysconfig/network-devices/ifconfig.<network interface>/ipv4*. A DNS servery v souboru */etc/resolv.conf*.
- Skript dále vytvoří soubor */etc/fstab*, kde je třeba nastavit adresu root disku v /dev/, jeho filesystém a adresu swapu v */dev/*.
- Dalším krokem je nastavení kernelu a jeho kompilace. Pokud jde pouze o to vyzkoušet LFS (pozn. musí být použit filesystém ext3, ext4, je třeba zaškrtnout v sekci File systems), je možné použít defaultní config. Tedy v menu, které vyskočí zvolit <exit>. Jinak je nastavení kernelu komplexní a složitou záležitostí, které je třeba věnovat čas, uvážit naše potřeby a HW konfiguraci zařízení na němž systém poběží. [4]

- Posledním krokem, ke kterému skripty navedou, je pak nastavení programu grub. Uživatel musí zadat */dev* adresu disku, na kterém se nachází LFS např. sdb sda. GRUB, poté vygeneruje soubor */boot/grub/grub.cfg* potřebný pro zavedení jádra systému.

Nyní je již třeba postupovat dle knihy, otestovat nastavení programu GRUB a aktualizovat Master Boot Record. Tyto kroky však nejsou třeba, pokud se chystá uživatel provozovat LFS vedle jiného systému, v tomto případě stačí pouze přidat položku LFS do souboru nastavení programu GRUB ve stávajícím systému.

Test GRUBu: Pro test je třeba, aby byl v hostovském systému nainstalován GRUB Legacy.
Nejprve je třeba restartovat počítač. A poté v příkazovém řáďku GRUBu nahrát obraz GRUBu z LFS.

```
/sbin/reboot
grub> root (hd0,1)
grub> kernel /boot/grub/core.img
grub> boot
```

Měl by naběhnout GRUB z LFS.

Posledním krokem je pak opět v chrootu LFS nahrát MBR (Master boot record) na disk s LFS. Je třeba nastavit diskovou adresu v */dev* např. */dev/sda /dev/sdb*, opět bez čísla oddílu.

```
./scripts/chap6/restart
./scripts/chrootIN/chrootAfterChap6
grub-setup /dev/sda
```

Teď by po restartu měl naběhnout bootloader z LFS a LFS by měla v pořádku naběhnout.

9.2.3 Instalace xorg

Malým bonusem v balíku skriptů LFS by bash scripts je pak skript pro instalaci grafického Xserveru (dále jen Xserver). K jeho spuštění je třeba nejprve v chrootu sytému nainstalovat balíček wget.

```
./scripts/BLFS/wget
```

Po instalaci wget je už možné přímo z LFS nainstalovat Xserver. Pokud je třeba změnit nějaké proměnné, např. proměnou WAIT, která určuje zda se má skript pozastavit v každém kroku, je možné toto provést v souboru */scripts/BLFS/variables*.

```
./scripts/BLFS/xorg
```

Skript končí naběhnutím testu Xserveru a vytvořením konfiguračního souboru *xorg.conf*. Je třeba přesunout konfigurační soubor do */etc/X11*, vytvořit soubor *.xinitrc*, který říká Xserveru jaké aplikace při startu spustit a vytvořit složku pro log soubor.

```
install -v -m644 ~/xorg.conf.new /etc/X11/xorg.conf
cat > ~/.xinitrc << "EOF"
# Begin .xinitrc file
xterm  -g 80x40+0+0   &
xclock -g 100x100-0+0 &
twm
EOF
cat >> /etc/sysconfig/createfiles << "EOF"
/tmp/.ICE-unix dir 1777 root root
EOF
```

Nyní už zbývá jenom spustit grafický server, který naběhne do správce oken twm.

Někdy se stane, zvláště u systému ve VirtualBoxu, že Xserver skončí chybou (Chyba nebývá na úplné konci výpisu, ale vně. Proto je lepší si chybu nechat vypsat samostatně):

```
cat /var/log/Xorg.0.log | grep EE
...
(EE) open /dev/fb0: No such file or directory
(EE) Screen(s) found, but none have a usable configuration.
```

Taková chyba je způsobena nastavením driveru Xserveru je třeba v souboru */etc/X11/xorg.conf* změnit v sekci Device položku Driver z hodnoty fbdev na vesa.

```
Section "Device"
  Driver "vesa"
EndSection
```

Bohužel příčin pádu Xserveru může být spousta v závislosti na HW a jiných okolnostech. Nejjednodušší způsob jak najít důvod chyby a její opravu, je zadat výpis chyby do internetového vyhledávače.

9.2.4 Onlyunpackscript

Tento skript slouží pouze pro stažení, kontrolu a rozbalení balíčků. Před jeho použitím je třeba mít vytvořený a připojený diskový oddíl jak je popsáno (viz. kapitola 9.2.2/s54).

Je třeba stáhnout skripty a překopírovat na místo

```
cd ~/
git clone git://github.com/zajca/LFS---by-bash-scripts.git
# mkdir $LFS/scripts
# cp -R ~/LFS---by-bash-scripts/* $LFS/scripts/
```

Je třeba nastavit 3 proměnné:

```
export ROOT=/mnt/LFS
export WAIT=1
export UNPACK=1
```

ROOT určuje přípojný bod disku s LFS

WAIT určuje zda se má při každém kroku čekat, až uživatel potvrdí další krok

UNPACK určuje, zda se má každý balíček rozbalit

Kombinací proměnných je tak možné dosáhnout pouze stažení všech balíčků bez čekání nebo stažení balíčků a jejich rozbalení s čekáním na další krok. Pokud chceme sestavovat LFS manuálně podle knihy, je dobré nastavit proměnné tak, jak je uvedeno v poli s kódem. Balíček se stáhne rozbalí, a skript čeká, až uživatel balíček sestaví a stiskne klávesu ENTER. Pokud při sestavení nastane chyba a je balíček třeba znovu sestavit, stačí zadat libovolný vstup a stisknout ENTER, balíček se znovu rozbalí. **!!!POZOR. Pokud se balíček sestavuje odděleným buildem, tedy ve složce balíček-build, je třeba tuto složku smazat manuálně!!!**

9.2.5 Pomoc při selháních

Protože pravděpodobnost sestavení LFS bez jakéhokoliv problému není nikterak velká, existuje několik míst, kde hledat při problému pomoc.

Google je prvním místem kam by se měl uživatel podívat, je relativně malá pravděpodobnost, že chyba která nastala, nenastala někomu již dříve.

irc channel LFS pro některé uživatele je IRC mrtvé, někteří o něm nikdy ani neslyšeli. V každém případě je IRC možností nejrychlejší odezvy. V IRC channelu LFS je téměř vždy kolem 30 uživatelů ochotných pomoci. **Ale první je třeba vždy použít google!**
Nastavení IRC clienta:
server: irc.linuxfromscratch.org
channel: #lfs-support
pro obecné dotazy existuje channel: #lfs

Fórum pro dotazy existuje také oddělení fóra LinuxQuestions.org speciálně pro LFS lze ho nalézt na adrese: http://www.linuxquestions.org/questions/linux-from-scratch-13/

mailing-list pro spoustu uživatelů také neznámá sféra, ale co se týká linuxové komunity asi nejpoužívanější je mailing-list. Mailing-list funguje úplně jednoduše, stačí poslat email na adresu mailing-listu a ten je následně zařazen mezi ostatní dotazy. Je dobré přihlásit se pro odběr novinek na adrese:
http://linuxfromscratch.org/mailman/listinfo/lfs-support/.
Adresa pro dotazy je pak: lfs-support@linuxfromscratch.org

9.3 Možnosti vylepšení scriptů

Tak jako u všech programů existuje mnoho možností vylepšení i u vytvořených skriptů pro sestavení LFS.

Možnosti vylepšení:

- Částečná změna skriptů z jazyka BASH na scripty programu GNU/make (tzv. Makafiles). Řídící skript makeall a onlyunpackscript by se po změně na makefile staly lépe ovladatelnými, především při přerušení a obnově práce. Případně použít menuconfig jako používá např. jádro linuxu, pro nastavení co se má kompilovat.
- Lepší přechody mezi login do chroot tak, aby nebylo potřeba přerušit skript, toto by šlo vyřešit přes program **sudo**, ale ten často v systémech pro build není dostupný.
- Přidání wget do toolchainu, tento krok by zajistil, že by nebylo třeba stahovat balíčky před začátkem sestavení, ale wget má další závislosti, které by bylo třeba vyřešit a pravděpodobně by bylo třeba jinak zkompilovat překladač v toolchainu.
- Další zásadním zlepšením by bylo vytvoření jakéhosi balíčkovacího systému. Zvláště u instalací programů z BLFS už je třeba hlídat závislosti na jiných balíčcích a často se může stát, že je některý program zbytečně kompilován dvakrát. Balíčkovací systém by mohl být založen na principu, že pro každý nainstalovaný balíček je vytvořen soubor nebo složka např. ve */var/packages* se stejným názvem jako balíček. Takový soubor by pak nesl informace o tom, o jaký program se jedná, jaké má závislosti a jaké soubory nainstaloval do systému (Pokud by v základu neuměl takový systém mazat nainstalované balíčky, nebylo by třeba seznamu souborů). Taková věc by se sebou nesla i změny ve skriptech, bylo by třeba, aby skript obsahoval i seznam závislostí nejlépe v poli, aby se dalo s těmito informacemi lépe pracovat a jednotlivé části skriptu by se rozdělily do funkcí.

ZÁVĚR

Cílem této bakalářské práce bylo vytvořit sadu skriptů v jazyce BASH pro automatické sestavení Linuxové distribuce Linux From Scratch. Pro návrh skriptů byl použit návod pro sestavení distribuce Linux From Scratch a nemalou měrou přispěl i verzovací systém GIT a komunitní web pro správu git repozitářů GitHub.

V teoretické části bylo provedeno seznámení s Linuxem, jeho distribucemi a možnostmi tvorby distribucí včetně kompilace zdrojových kódů a programování v interpretovaném jazyce BASH, což jsou důležité znalosti pro vytvoření distribuce Linuxu. Dále bylo provedeno seznámení se zabezpečením Linuxu a systémem hodnocení bezpečnosti Softwaru. Praktická část byla věnována skriptům v interpretovaném jazyce BASH, jejich struktuře, obsahu a návodu pro použití. Skripty byly otestovány jak na virtuálním počítači, vytvořeného programem VirtualBox a s použitím systému Ubuntu 10.10 Maverick Meerkat, tak na fyzickém počítači se systémem ArchLinux.

Programování v interpretovaném jazyce BASH je velmi zajímavé, zvláště vzhledem k jeho integraci se systémem Linux. Jsem rád, že jsem si mohl rozšířit prostřednictvím této bakalářské práce znalosti tohoto jazyka, o tomto svědčí i fakt, že jsem dvakrát přepsal způsob jakým skripty fungují. Také jsem rozšířil své znalosti o tom jak funguje Operační systém Linux v širším měřítku, za což jsem velmi rád.

ZÁVĚR V ANGLIČTINĚ

The aim of this bachelor work was to create a set of BASH scripts for automatic compilation Linux From Scratch distribution. For the design of scripts was used Linux From Scratch book and in no small part contributed GIT version control system and community site for managing git repositories GitHub.

In the theoretical part was made introduction with Linux, the Linux distributions and creation of Linux distributions, including a compilation of source code and programming interpreted language BASH, which are important skills for a building Linux distribution. It was also made familiar with the security of Linux and system software safety assessment. The practical part was devoted to scripts in BASH, its structure, contents and instructions for use. Scripts was tested in virtual computer created by program VirtualBox and runing Ubuntu 10.10 Meerkat Maverick, and at the physical computer running ArchLinux.

Programming in interpreted language BASH is very interesting, especially due to integration with Linux. I'm glad that I was able to extend my knowledge of BASH becouse of this work. True of that was demonstrated by the fact that I'm twice rewrote the way how the script works. I also expanded my knowledge about the functioning of the Linux operating system on a larger scale, for which I am very happy.

SEZNAM POUŽITÉ LITERATURY

[1] COOPER, Mendel. *Advanced Bash-Scripting Guide* [online]. Revision 6.2. [s.l.] : [s.n.], 17 March 2010, Dostupné z WWW: http://www.tldp.org/LDP/abs/html/index.html. [e-kniha]

[2] MILAR, Bohdan. *Bash očima Bohdana Milara* [online]. [s.l.] : LinuxExpres, 2008. Dostupné z WWW: http://www.root.cz/knihy/bash-ocima-bohdana-milara/. [e-kniha]

[3] BEEKMANS, Gerard. *Linux From Scratch* [online]. 6.7. [s.l.] : [s.n.], 1999, 2010. Dostupné z WWW: http://www.linuxfromscratch.org/lfs/view/stable/. [e-kniha]

[4] KROAH-HARTMAN, Greg. *Linux kernel in a nutshell* [online]. [s.l.] : O Reilly Media, Inc., 2006. Dostupné z WWW: http://www.root.cz/knihy/linux-kernel-in-a-nutshell/. [e-kniha]

[5] KOLEKTIV, Autorů, et al. *Linux: Dokumentační projekt* [online]. 4. vyd. Brno : Computer Press, 2008. Dostupné z WWW: http://www.root.cz/knihy/linux-dokumentacni-projekt-4-vydani/. ISBN 978-80-251-1525-1. [e-kniha]

[6] LITERÁK, Leoš. *Co je to Linux. AbcLinuxu.cz* [online]. 17.1.2006, [cit. 2011-03-30]. Dostupné z WWW: http://www.abclinuxu.cz/ucebnice/uvod/co-je-to-linux.

[7] BEEKMANS, Gerard. *Linux From Scratch* [online]. 1998 [cit. 2011-02-04]. Linux From Scratch. Dostupné z WWW:http://www.linuxfromscratch.org/lfs/.

[8] MARTINEK, David. *Fakulta Informačních Technologií VUT* [online]. 23. září 2010 [cit. 2011-03-30]. Překlad programu. Dostupné z WWW: http://www.fit.vutbr.cz/ martinek/clang/gcc.html.

[9] *Nette Framework* [online]. 2011 [cit. 2011-03-30]. Český překlad nové BSD licence. Dostupné z WWW: http://nette.org/cs/licence/newbsd.

[10] *Linuxové distribuce. In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-02-23]. Dostupné z WWW: http://cs.wikipedia.org/wiki/Linuxové_distribuce.

[11] *The Debian GNU/Linux FAQ* [online]. 2009 [cit. 2011-03-30]. Dostupné z WWW: http://www.debian.org/doc/FAQ/ch-pkgtools.en.html.

[12] *Dokumentace Gentoo Linuxu* [online]. 2006 [cit. 2011-03-30]. Dostupné z WWW: http://www.gentoo.org/doc/cs/handbook/handbook-x86.xml?part=2&chap=1.

[13] *ABCLinuxu* [online]. 13.11.2010 [cit. 2011-04-29]. Bash. Dostupné z WWW: http://www.abclinuxu.cz/software/programovani/jazyky/bash.

[14] FUCHS, Jan. *ABCLinuxu* [online]. 2003 [cit. 2011-04-30]. Seriál: BASH. Dostupné z WWW: http://www.abclinuxu.cz/serialy/bash.

[15] SHARMA, Mayank. *TechRadar UK* [online]. 2010-1-24 [cit. 2011-05-01]. 10 scripts to create your own Linux distribution. Dostupné z WWW: http://www.techradar.com/news/software/operating-systems/10-scripts-to-create-your-own-linux-distribution-665247.

[16] *InternetNews* [online]. March 19, 2001 [cit. 2011-05-08]. German Army No Longer Uses Microsoft Programs. Dostupné z WWW: http://www.internetnews.com/bus-news/article.php/716871/Report-German-Army-No-Longer-Uses-Microsoft-Programs.htm.

[17] HOCHMUTH, Phil. *Network world* [online]. January 19, 2004 [cit. 2011-05-08]. Linux delivers for U.S. Postal Service. Dostupné z WWW: http://www.networkworld.com/techinsider/2004/0119linuxcase.html.

[18] Free Software Foundation. *GNU Operating System* [online]. 1996 [cit. 2011-05-09]. Dostupné z WWW: http://www.gnu.org/.

[19] *ABCLinuxu* [online]. 2004, 2009 [cit. 2011-05-09]. Gnu - výkladový slovník. Dostupné z WWW: http://www.abclinuxu.cz/slovnik/gnu.

[20] KRČMÁŘ, Petr. *Root.cz* [online]. 2011-05-09 [cit. 2011-05-09]. Proč není Hurd ani po dvaceti letech hotový?. Dostupné z WWW: http://www.root.cz/clanky/proc-neni-hurd-ani-po-dvaceti-letech-hotovy/.

[21] VIRIJEVICH, Paul. *Linux.com* [online]. February 15, 2005 [cit. 2011-05-10]. Securing Linux with Mandatory Access Controls. Dostupné z WWW: http://www.linux.com/archive/feature/113941.

[22] The SCO Group, Inc. *The SCO group, inc.* [online]. 22 April 2004 [cit. 2011-05-10]. The root account and system owner. Dostupné z WWW: http://uw714doc.sco.com/en/HANDBOOK/saN.superuser.html.

[23] KUNDEROVÁ, Ludmila. *Hodnocení informační bezpečnosti* [online prezentace]. Brno : Ústav informatiky, PEF MZLU, [cit. 2011-05-14]. Dostupný z WWW: https://akela.mendelu.cz/ lidak/share/snimky-bis/prednaska5.ppt.

[24] *Linux Documentation Project* [online]. 2004-08-16 [cit. 2011-05-14]. Linux dictionary. Dostupné z WWW: http://www.tldp.org/LDP/Linux-Dictionary/html/t.html.

[25] Narodní bezpečnostní úřad. *INFORMACE O HODNOCENÍ BEZPEČNOSTI INFORMAČNÍCH TECHNOLOGIÍ* [online]. Praha : NBÚ, 2005 [cit. 2011-05-14]. Dostupné z WWW: http://www.nbu.cz/_downloads/bezpecnost-informacnich-systemu/container-nodeid-748/infoobit.pdf.

[26] *NIAP CCEVS* [online]. 2007 [cit. 2011-05-14]. Validated Product - Red Hat Enterprise Linux Version 5 running on IBM Hardware. Dostupné z WWW: http://www.niap-ccevs.org/cc-scheme/st/?vid=10125.

[27] *NIAP CCEVS* [online]. 2007 [cit. 2011-05-14]. Validated Product - SUSE Linux Enterprise Server 10 SP1. Dostupné z WWW: http://www.niap-ccevs.org/st/vid10271/

[28] *SElinux Wiki* [online]. 2007 [cit. 2011-05-16]. Main Page. Dostupné z WWW: http://selinuxproject.org/page/Main_Page.

[29] *AppArmor* [online]. 2006 [cit. 2011-05-16]. Main Page. Dostupné z WWW: http://wiki.apparmor.net/index.php/Main_Page.

[30] NTT DATA Corporation. *TOMOYO Linux Home Page* [online]. 2006 [cit. 2011-05-16]. TOMOYO Linux. Dostupné z WWW: http://tomoyo.sourceforge.jp/.

[31] WILLIAMS, Jeff. *OWASP* [online]. 2006 [cit. 2011-05-16]. Null-pointer dereference. Dostupné z WWW: https://www.owasp.org/index.php/Null-pointer_dereference.

[32] *Grsecurity* [online]. 2004 [cit. 2011-05-16]. Grsecurity. Dostupné z WWW: http://grsecurity.net/.

[33] *The netfilter.org project* [online]. 1999 [cit. 2011-05-16]. Netfilternetfilter/iptables project homepage. Dostupné z WWW: http://www.netfilter.org/.

[34] StatCounter. *StatCounter Global Stats* [online]. 1999 [cit. 2011-05-16]. Browser, OS, Search Engine including Mobile Market Share. Dostupné z WWW: http://gs.statcounter.com/#os-ww-monthly-201004-201104-bar

[35] E-Soft Inc. *Web Server Survey* [online]. 1998, 2010-05-01 [cit. 2011-05-16]. SecuritySpace. Dostupné z WWW: http://www.securityspace.com/s_survey/data/201004/index.html.

[36] ClamAV. *Clam Antivirus* [online]. 2002 [cit. 2011-05-16]. Clam Antivirus. Dostupné z WWW: http://www.clamav.net/lang/en/.

[37] OHNESORG, Dan. *Linux.cz* [online]. 2007 [cit. 2011-05-16]. Linux a viry. Dostupné z WWW: http://www.linux.cz/viry.html.

[38] BARRETT, PH. D., Daniel J.; SILVERMAN, Richard E.; BYRNES, Robert G. *SSH: The Secure Shell* [online]. 2001, 2010 [cit. 2011-05-16]. The Definitive Guide. Dostupné z WWW: http://www.snailbook.com/index.html.

[39] WEIMER, Florian. *Debian* [online]. 2008-05-13 [cit. 2011-05-16]. [SECURITY] [DSA 1571-1] New openssl packages fix predictable random number generator. Dostupné z WWW: http://lists.debian.org/debian-security-announce/2008/msg00152.html.

[40] *ArchWiki* [online]. 2009-06-03 [cit. 2011-05-16]. Pacman package signing. Dostupné z WWW: https://wiki.archlinux.org/index.php/Pacman_package_signing.

[41] *FedoraProject* [online]. 2008-05-24 [cit. 2011-05-17]. How to create and use a Live CD. Dostupné z WWW: http://fedoraproject.org/wiki/FedoraLiveCD/LiveCDHowTo.

[42] BAUMANN, Daniel. *Debian Live* Project [online]. 2010 [cit. 2011-05-17]. Debian Live Project. Dostupné z WWW: http://live.debian.net/.

[43] *ArchWiki* [online]. 2005-07-23 [cit. 2011-05-17]. Building a Live CD. Dostupné z WWW: https://wiki.archlinux.org/index.php/Building_a_Live_CD.

[44] *GoboLinux* [online]. 2006 [cit. 2011-05-17]. What is GoboLinux?. Dostupné z WWW: http://www.gobolinux.org/index.php?page=at_a_glance.

[45] *GoboLinux* [online]. 2006-05-23 [cit. 2011-05-17]. The GoboLinux Filesystem Hierarchy. Dostupné z WWW: http://wiki.gobolinux.org/index.php?title=The_GoboLinux_Filesystem_Hierarchy.

[46] STRAUSS, Daryll. *Linux Journal* [online]. 1998-01-01 [cit. 2011-05-17]. Linux Helps Bring Titanic to Life. Dostupné z WWW: http://www.linuxjournal.com/article/2494.

[47] OWE, Robin.*Linux Journal* [online]. 2007-06-05 [cit. 2011-05-17]. DreamWorks Animation "Shrek the Third": Linux Feeds an Ogre. Dostupné z WWW: http://www.linuxjournal.com/article/9653.

[48] JAQUES, Robert. *IT News from V3.co.uk* [online]. 2004-06-29 [cit. 2011-05-17]. Linux behind the magic of Shrek 2. Dostupné z WWW: http://www.v3.co.uk/v3-uk/news/1972166/linux-magic-shrek.

[49] SIMPSON, Stacy. *Linux for Devices* [online]. 2001-10-11 [cit. 2011-05-17]. IBM & Citizen Watch develop Linux-based "WatchPad". Dostupné z WWW: http://www.linuxfordevices.com/c/a/News/IBM-Citizen-Watch-develop-Linuxbased-WatchPad/.

[50] *GNewSense GNU/Linux* [online]. 2006-08-27 [cit. 2011-05-17]. Builder/HowToCreateYourOwnGNULinuxDistribution. Dostupné z WWW: http://www.gnewsense.org/Builder/HowToCreateYourOwnGNULinuxDistribution.

[51] HARTL, Michael. *Ruby on Rails Tutorial* [online]. [s.l.] : [s.n.], 2010 [cit. 2011-05-21]. Dostupné z WWW: http://ruby.railstutorial.org/book/ruby-on-rails-tutorial#sec:secure_password_theory.

[52] PALÁT, Pavel. *LINUXZONE* [online]. 2004-05-07 [cit. 2011-05-25]. Grsecurity. Dostupné z WWW: http://www.linuxzone.cz/index.phtml?ids=1&idc=1060.

[53] TORVALDS, Linus. *LKML.ORG - the Linux Kernel Mailing List Archive* [online]. 2005-04-07 [cit. 2011-05-26]. Linus Torvalds: Re: Kernel SCM saga. Dostupné z WWW: http://lkml.org/lkml/2005/4/8/9.

[54] CHACON, Scott. *Git* [online]. 2005 [cit. 2011-05-26]. Git - Fast Version Control System. Dostupné z WWW: http://git-scm.com/.

[55] GitHub Inc. *Help.GitHub* [online]. 2011 [cit. 2011-05-26]. Set Up Git (Linux). Dostupné z WWW: http://help.github.com/linux-set-up-git/.

[56] Linux Kernel Organization, Inc. *The Linux Kernel Archives* [online]. 2006 [cit. 2011-05-26]. Git User's Manual (for version 1.5.3 or newer). Dostupné z WWW: http://www.kernel.org/pub/software/scm/git/docs/user-manual.html#how-to-merge.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

LFS	Linux From Scratch
OS	Operační systém
HW	Hardware
SW	Software
GPL	General Public License
BSD	Berkeley Software Distribution
EULA	End-User-License-Agreement
RHEL	Red Hat Enterprise Linux
SLED	SUSE Linux Enterprise Desktop
Např	například
APT	Advanced Packaging Tool
DPKG	Debian package management system
GUI	Graphic user interface
RPM	Red Hat Package Manager
GNU	GNU is Not Unix
MAC	Mandatory Access Control
DAC	Discretionary Access Control
RBAC	Role-Based Access Control
TCSEC	Trusted Computer System Evaluation Criteria
ITSEC	IT Security Evaluation Criteria
CTCPEC	Canadian Trusted Computer Product Evaluation Criteria
CC	Common Criteria for Information Technology Security Evaluation
FC	Federal Criteria
CCRA	Common Criteria Recognition Arrangement
EAL	Evaluation Assurance Levels
SElinux	Security-Enhanced Linux
LSM	Linux Security Modules
IPC	interprocess communication
SSH	Secure Shell
PP	Protection Profile
TOE	Target of Evaluation
ST	Security Target
SUID	secure user identification number
SGID	secure group identification number
EUID	effective user identification number
GPG	GNU Privacy Guard

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I. PDF VERZE BAKALÁŘSKÉ PRÁCE A ZDROJOVÝ KÓD V JAZYCE LATEX

PDF je dostupné na DVD v portále STAG a na internetu

https://github.com/zajca/BC.TvorbaLinuxoveDistribuce

Spolu se zdrojovým kódem práce je dostupná i upravená šablona pro FAI UTB. Pro přeložení je třeba překladač XeLaTeX.

Na DVD dostupné ve složce *BC.TvorbaLinuxoveDistribuce*. BC.pdf je výstupem ve formátu PDF. BC.tex je zdrojovým kódem v jazyce LaTeX. A ve složce *sablona* se nachází soubor *sablona.tex* jež je šablonou pro FAI UTB.

PŘÍLOHA P II. ZDROJOVÉ KÓDY LFS BY BASH SCRIPTS

Zdrojové kódy pro automatické sestavení distribuce LFS.

Dostupné na DVD (složka *LFS—by-bash-scripts*) na portále stag a na internetu na adrese

https://github.com/zajca/LFS—by-bash-scripts

Návod na použití skriptů (viz. kapitola 9.2.2/s54) nebo anglicky v README souboru projektu.

Na portále STAG a na DVD dostupná verze BCv6.8. Verze dostupná na internetu se může lišit, návod jak se dostat k verzi BCv6.8 (viz. kapitola 9.2.2/s54)

PŘÍLOHA P III. FUNKČNÍ TOOLCHAIN PRO 64BIT SYSTÉM

Dostupné pouze na DVD jako soubor toolchain64.tar.gz (148,970,636 bytes)

Funkční toolchain pro sestavení LFS na 64bitovém procesoru. Jedná se o soubory, po dokončení kapitoly 5 knihy LFS, tedy o obsah složky /tools.

PŘÍLOHA P IV. FUNKČNÍ SESTAVENÍ DISTRIBUCE LFS VČETNĚ X-SERVERU

Dostupné pouze na DVD jako soubor LFS.ova (2,573,794,816 bytes)

Jedná se o již nastavený virtuální počítač pro zdarma dostupný program VirtualBox. Stačí pouze otevřít VirtualBox a spustit *FILE -> Import Appliance*. Velmi jednoduchý wizard pak nahraje vše potřebné a přidá položku LFS.

SW vybavení: Grafický server Xorg, window manager PekWM, programovací jazyk RUBY s nainstalovaným gemem ShowOff pro prezentace, webový prohlížeč opera.

Přihlášení:
Uživatel: root
Heslo: 1rootlfs

Spuštění grafického serveru příkazem *xinit*.