Tvorba distribuce OS Linux

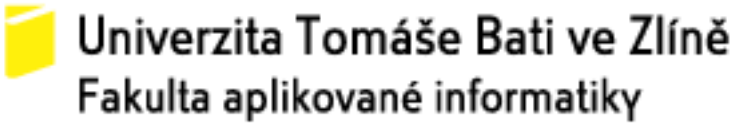
Building Linux distribution

Martin Zajíc

Bakalářská práce
2011

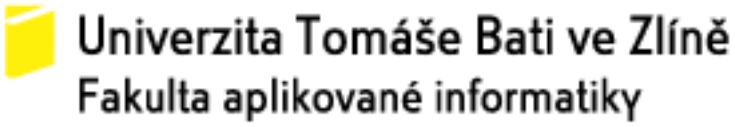
Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

nascannované zadání s. 1



Obr. 1. logo

nascannované zadání s. 2



Obr. 2. logo

ABSTRAKT

Klíčová slova:

ABSTRACT

Keywords:

poděkování, motto, úryvky knih, básní atp.

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

text

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 tučnáka 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 predstavovaly poslední bariéru, kterou bylo nutno překonat. V 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,…), kopatibilní s aplikacemi Microsoft Office (LibreOffice, Koffice, AbiWord,…) [5]. Co se týče použití na serverech, 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 internetoveho 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 vykonávají [5].
Stojí také za zmínku, že moderní Linux běží nejen na pracovních stanicích, středně 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 ikdyž jsou často milně spojovány pouze s Linuxem. Kromě Linuxových distribucí existují například distribuce s jádrem BSD (OpenBSD, NetBSD, FreeBSD,…), HURD (ArchHurd), XNU (PureDarwin) a mnoho jiných. Přídávný SW distribuce je čistě volbou autora distribuce, může se tak jednat například o SW projektu GNU (nejvýznamějšími komponenty jsou GCC, glibc, Emacs nebo GNOME) [18], SW projektu BSD, proprietální (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 vydavaný v rámci projektu GNU je licencován GNU General public license (GPL). Dodnes je vyvýjeno původní microkernel jádro GNU/HURD, avšak ani za 21let 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 ohledech. Nejvýraznějšími rozdíly, které dělají distribuce distrbucemi 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 jejich odstranění je prohlášena za stabilní a balíčky v ní obsažené jsou přesunuty do větve *stable* [1)]. RHEL vychází v nepravidelných cyklech (cca 2-3roky) a v mezičase vychází updaty. [2)]

Krátký cyklus vydání Typicky půl roku až rok. Používá ho většina desktopových 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 včele 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 (narozdíl 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 založaných 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 balížků 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) kdyžto u RPM distribucí se setkáme s repozitáři zvláš pro multimedia, core system, apd...

PACMAN Jedná se pravděpodobně o jeden s nejrychlejších balíčkovacích systémů a je součástí distribuce ArchLinux jejích derivátů larch, FaunOS, Archie, Chakra

[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 větvy putuje do větve *testing*

[7] APT je několik aplikací (apt-cache, apt-get,…), které každý poskytují jinou funkci

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 script 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 naráz aby byl balíčet 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 na, které 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 vsechny níže zmíněné zařízení.

Desktop Dalšími zařízeními na 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 narozdíl 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 jinym 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 najde i v embedded zařízeních jako jsou settopboxy, televize, IP kamery, pračky, mikrovlnky, ledničky a jiné jednoúčelové zařízení,…

Mobilní Asi nejnovějším prostředí ve, kterém momentálně již dominuje systém linux 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 ikdyž 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 oběvují 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 softwarove výbavě. **Ubuntu, Kubuntu, Edubuntu, Xubuntu, Gobuntu, Lubuntu, Fluxbuntu, Ubuntu Netbook Edition**

Velký rozdíl zavádí do linuxových ditribucí **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 otravovat 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. 1. Logo Linux From Scratch

- LFS učí uživatele jak Linux funguje uvnitř.
- Sestavování LFS učí o všem co v systemu běží, jak jednotlivé pracují a závisí jedna na druhé. A jak upravit systém podle chuti.
- Sestavením LFS získáme velice kompaktní Linuxový systém
- Pokud instalujete běznou Linuxovou distribuci, nainstalujete i množství programů, které pravděpodobně nikdy nepoužijete. LFS jde sestavit i ve velikosti nepřesahující 100MB
- LFS je extrémě flexibilní. Máte moc systém sestavit přesně podle vašich potřeb.
- LFS můžete zabezpečit na míru. Nemusíte čekat, jako u binárních distribucí, až někdo sestaví balíček s bezpečtnostním patchem, který potřebujete. Můžete se 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 nástroje pro autmatické sestavování LFS a BLFS

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

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 dokumnetů, které popisují jak vylepšit vaše sestavení LFS jinak než je popsáno v LFS a BLFS.

LiveCD projekt jehož hlavním cílem je vytovřit LiveCD vhodné jako hostotitelský 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 script 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 script 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 možno 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 urovni"
$ export TRI
$ export
declare -x DVA="Proměnná exportovaná do podřízeného shellu"
declare -rx TRI="Proměnná pouze pro čtení, ale jen na lokální urovni"
$ readonly
declare -rx TRI="Proměnná pouze pro čtení, ale jen na lokální urovni"
$ 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 přídá 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

4.2 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 uživately";
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 uživately";
fi
```

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

```
#testujeme dvě podmínky
if [ $USER == "root" ] && [ $LANG == "cs_CZ" ]; then
  echo "Jsi český admin"
fi
#pokud funkce skončí chybou script 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 cokoliv 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 používájí stejným způsobem ve scriptech 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 odělují 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 souborech rozličných formátu 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 programátorovy jestly 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 neastěji 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 standartně 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 stejnojmeného 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 informce 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 danná 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 nasteveními pro překlad. Jedná se v zásadě o BASH script, 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 mám program nebo jeho části nainstalovat nebo také které části programu se mají přeložit*. Configure je scriptem BASHe a proto je také tak volána, výsledekem configure je vygenerovaní souboru, které řídí překlad.

5.2.6 Další soubory a alternativy některých souborů

5.3 Kompilace

Samotná kompilace sestává s 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 linuxoveho jádra to zahrnuje většinou použité patche.

[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

5.4.1 Vytvoření ze zdrojových kódů

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

1. Prve je třeba oddělit systém a vytvořit toolchain, který slouží k sestavení nové distribuce nezávislé na předchozím hostovském systému.
2. sestavení vlastní sady základních nástrojů pro pouzití systému. Sada obsahuje překladače, nástroje pro práci se soubory a jiné potřebné programove vybavení.
3. nastavení bootovacího procesu systému.
4. vytvoření tabulky souborových systémů, kompilace jádra systému, nastavení bootloaderu

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.

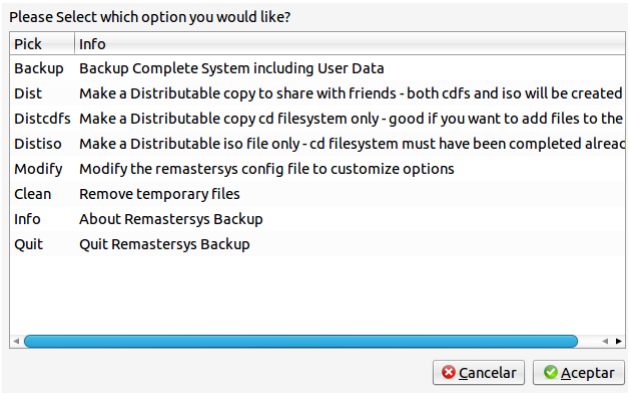
Remastersys

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 nástojem 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)
```

The Ubuntu Customisation Kit (UCK)

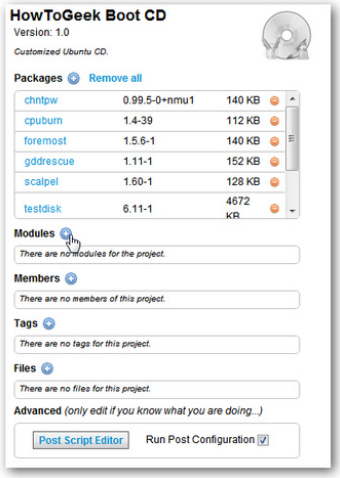
Program je určen pro Ubuntu a všechny jeho deriváty. Dovoluje odebrat nebo přidat jakékoliv aplikace na instalačním CD. UCK je určeno spíše pokročilejším uživatelům, protože po vytvoření LiveCD, do něj přihlásí uživatele prostřednictvím chroot a dovolí mu tak změnit prakticky cokoliv. Aplikace má jednoduché grafické rozhraní se spoustou dialogů, které vás provedou vytvořením LiveCD.[15]



Obr. 1. Grafické rozhraní programu Remastersys

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. 2. Webové rozhraní Reconstructoru

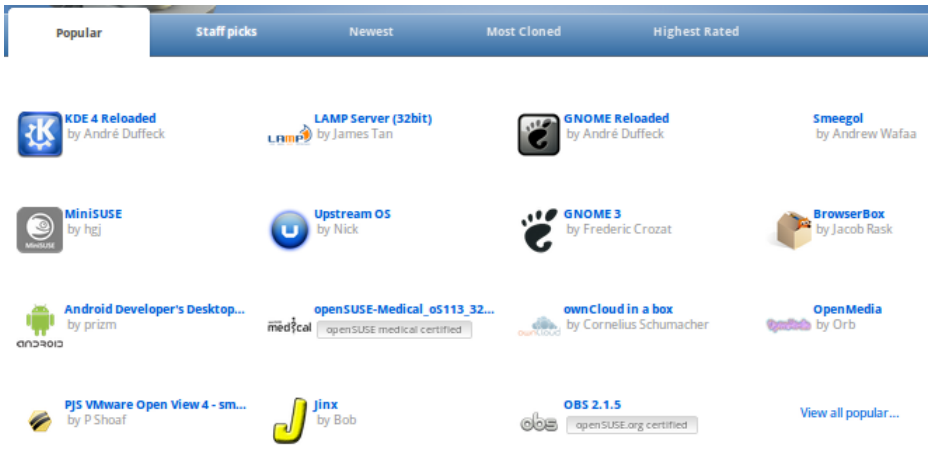
Revisor

Revisor je aplikace pro vytvoření vlastního instalčního 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]

SUSE Studio

Asi nejzajímavějším programem je webová aplikace od společnosti Novell, SUSE Studio. Nejenomže umí nakonfigurovat velké množství nastavení. Ale velice 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. 3. Grafické rozhraní programu Revisor



Obr. 4. Webové stránka s výběrem již existujích sestavení

Pungi

Pungi je nástrojem, který používají vývojári 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

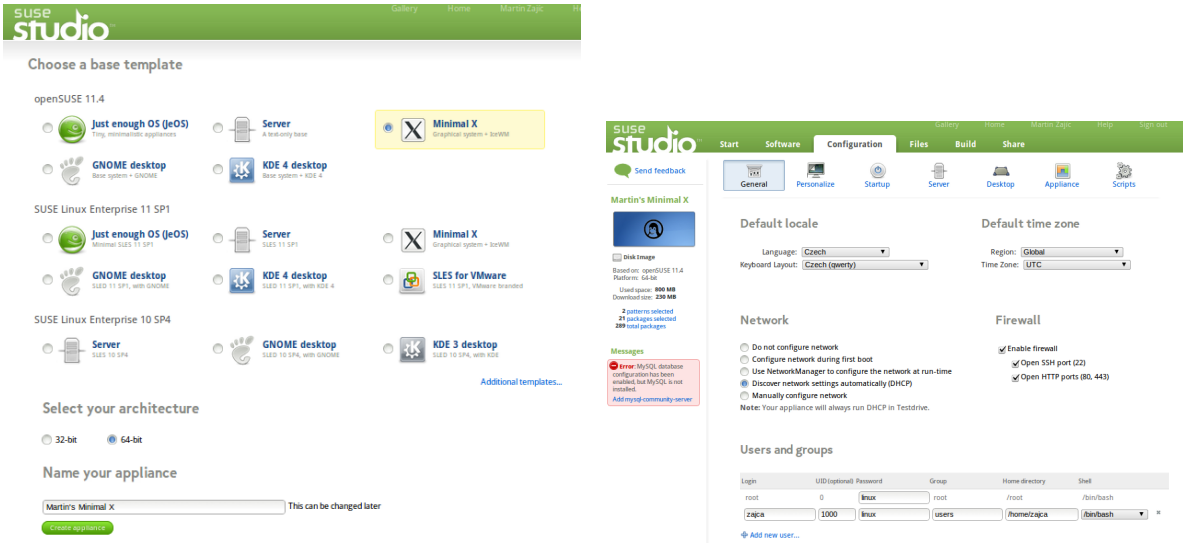
Builder je série scriptů napsaných v BASHi, které používají vývojáři derivátu distribuce Ubuntu "gNewSense"na sestavení jejich distribuce. Ke scriptům je také napsaný přehledný manuál jak je používat, který je dostupný online [15] [50].

MySlax Creator

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

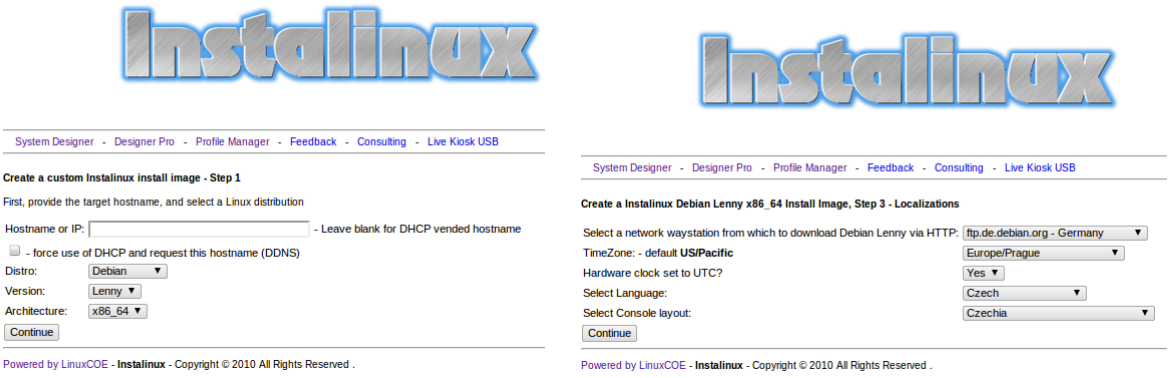
Instalinux

Jednoduchým, ale mocným nástrojem je Instalinux, umožňuje uživateli vybrat z několika distribucí (Debian, Ubuntu, CentOS, Fedora, OpenSUSE, Scientific) a v dalších



Obr. 5. Webové rozhraní s nastavením systému, v levo výběr základního rozhraní, v pravo podrobnější nastavení

krocích základní nastavení systému a výběr aplikací. Výsledkem je ISO obraz, kterým nemusí mít ani 30MB. Pro podrobnější nastavení exituje Designer Pro, který umožňuje nastavit mnohem více. Výbornou vlastností je uložení profilu, který je možné použít znovu a upravit si jej.

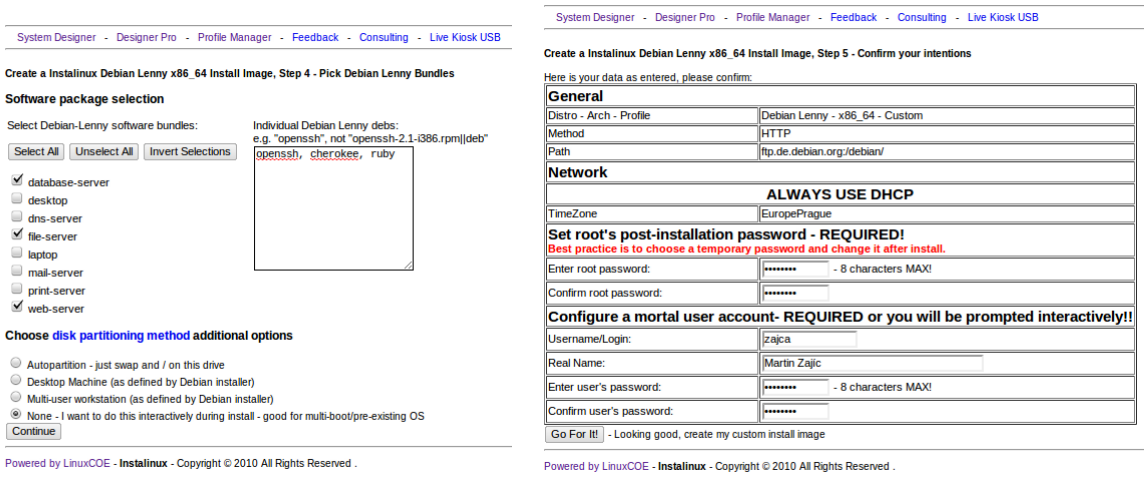


Obr. 6. Webové rozhraní Instalinux, v levo výběr distribuce, v pravo nastavení jazyka a časové zóny

Takto vytvořená distribuce může mít i 8MB!

6 ZABEZPEČENÍ LINUXOVÉHO SYSTÉMU

Zabezpečení OS je velice rozsáhlým a komplexním odvětvím, které se navíc velmi rychle vyvíjí a rozšiřuje. Jeho plné popsání by vadalo 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.



Obr. 7. Webové rozhraní Instalinux, v levo výběr aplikací, v pravo nastavení uživatelů

6.1 Access Control

Kontrola přístupu uvnitř systému je jedno z nejdůležitějších zabezpeční v OS vůbec. Určuje, který proces nebo uživatel má k čemu oprávnění. Existuje několik obecných modelů tohoto zabezpečení, v Linuxu se konkrétně využívá dvou modelů: **Discretionary Access Control (DAC)** a **Mandatory Access Control (MAC)**. Model zabezpečení DAC je bežnou součástí linuxového jádra a využívá jej defaultně většina distribucí. Model MAC je pak v linuxu dostupný díky různým rozšířením, jež nejčastěji využívají **Linux Security Modules (LSM)**, které zajišťující implementaci MAC v jádře.

6.1.1 DAC vs. MAC

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 spoštění (r+w+x), pro tento soubor. V tomto modelu uživatel kontroluje data dle vlastního uvážení. Vlastník systému[1)] nemá totální kontorlu nad systémem, tu má uživatel. [21]

Avšak největší hrozba v Linuxovém modelu je reprezentována účtem uživatele root. Tento super-uzivatel může kontorlovat všechny soubory a procesy. Pokud je účet uživatele root nebo proces s takto vysokými přístupovými právy kompromitovám, útoční získává kontorolu nad celým systémem a nad daty v něm. [21]

Bezpečnějším přístupem je omezit nebo přímo eliminovat účet uživatele root a přesunout tak práva z uživatelských účtů na na vlastníka systému[1)]. Tento přístup se nazývá **Mandatory Access Control (MAC)** [21]

MAC prosazuje bezpečnostní pravidla povinně (mandatory) narozdíl od DAC, který pravidla prosazuje volitelně (discretionary). Bezpečnostní pravidla může nastavit vlastník systému a implementovat systémový nebo bezpečnostní administrátor. Jakmile jsou pravidla jednou stanovena, žádný uživatel nemůže tyto pravidla měnit, ikdyby měl práva uživatele root. S bezpečnostní přístupem MAC je ochrana souborů a procesů nezávislá na na jejich vlastníkovy. [21]

[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 tyto ustanovení měnit.[22]

6.1.2 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 jiných zdrojů než jenom k souborům, např. přístum k síťovým zdrojům nebo k meziprocesové komunikaci (IPC).[28]

6.1.3 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 převším oproti SElinuxu vyznačuje mnohem jednoduším nastavením.

6.1.4 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 zahájem 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.

6.1.5 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 bezpečnotstní 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 jádra
- Omezení, která umožňuje uživateli zobrazit pouze své procesy
- Výstrahy zabezpečení a audity, které obsahují IP adresu osoby, která způsobila záznam

6.2 Hodnocní bezpečnosti OS a SW

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

6.2.1 Trusted Computer System Evaluation Criteria, TCSEC

Asi nejznáměší kniha popisující požadavky pro "důvěryhodný"operační systém, vydaná v roce 1985 americkým ministrestvem obrany (Department of Defense, DoD) a 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 6.2.4/s34)

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 poloformálního 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]

6.2.2 IT Security Evaluation Criteria,ITSEC

Evroská kritéria vytvořená v roce 1990 dnes již nahrazena krytérii Common Criteries (CC) (viz. kapitola 6.2.4/s34).
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

6.2.3 Canadian Trusted Computer Product Evaluation Criteria, CTCPEC

Kanadský soubor kriterií, který je dnes již také nahrazen Common Criteries (CC) (viz. kapitola 6.2.4/s34).

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

6.2.4 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 6.2.1/s32) a Federal Criteria, evropských ITSEC (viz. kapitola 6.2.2/s33) a kanadských CTCPEC (viz. kapitola 6.2.3/s34). 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]

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.

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.

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 procedura pro bezpečnou instalaci, generování a spouštění.

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.

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.

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.

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.

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.

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

6.3 Internetové zabezpečení

6.3.1 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 určným 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 které si uživatel vytvoří nastavení v grafického nebo textového rozhraní a výslednou konfiguraci firewallu pak přenese na server. Takovým nástrojem je např. **FirewallBuilder**, **FireHOL** oblíbená sada scriptů pro konfiguraci Netfilteru **Shorewall**.

Jak už je v linuxu dobrým zvykem vše je často přehledně a pouzitelně 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**.

6.3.2 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 jeho další 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.

6.3.3 SSH

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

Hlavní výhody:

- Množství autentizačních metod

- 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)

6.4 Hypotetické bezpečnostní hrozby v linuxu

Virové hrozby byli probrány (viz. kapitola 6.3.2/s36) 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 vyší oprávnění než jsou jí určeny, k samotnému přidělení opravnění 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 vytovořit 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 my si uživatel smazal disk. Samozřejmě takový kód je jen na ukázku, ale demostruje možnosti tohoto "útoku", daleko pravděpodobnější by bylo spuštění útočníkova scriptu 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 ve 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. 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.

7 LIVECD

Ač by se mohl zdát, že liveCD je nějaká velmi složitá věc k vytvoření není to tak úplně pravda. Nejjednoduší forma LiveCD je pokud strčíme CD do mechaniky, ten nabootuje a spustí se příkazová řádka ve které je možné pracovat. Takto fungovali a často ještě fungují instalační CD. Dnes se velmi často používá několik instalčních/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,…)

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

- 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. nahrání systému do RAM, pak už není třeba číst z CD nebo také pokud se jedná o LiveUSB jdou uložit změny v systému na disk a jsou načteny při dalším spuštění. (Slax,Chakra,…)

7.1 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 **nahraním celého systému do paměti RAM**, této proces má však velkou nevýhodu v délce trvání, protože celé LiveCD musí být přečteno a zapsáno do paměti RAM. Další nevýhodou je pak, až na vyjímky, nemožnost uložení změn v systému zpět na přenosné médium.
Samozřejmě LiveCD není většinou používáno k běžně práci, takže jeho výhoda ve vyzkoušení neznámého operačního systému nebo grafického prostředí bez nutnosti mnohdy zdlouhavé instalace, převišuje nad nevýhodami.

7.2 Možnosti tvorby LiveCD

Vytvořit LiveCD lze několika způsoby, nejjednoduší je způsob úpravy stávajícího LiveCD jak bylo i popsáno (viz. kapitola 5.4.2/s26).

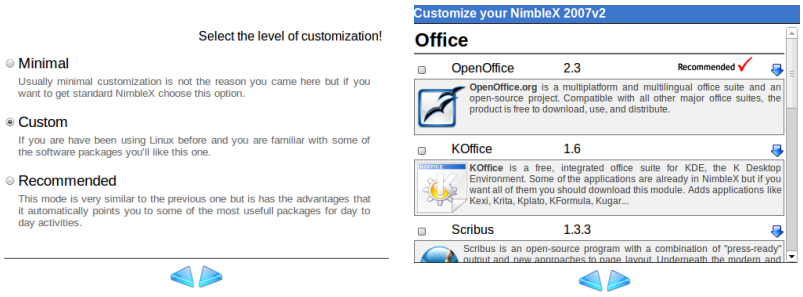
7.2.1 Obecný postup tvorby LiveCD

Obecný postup tvorby LiveCD se dá shrnout do několika kroků.

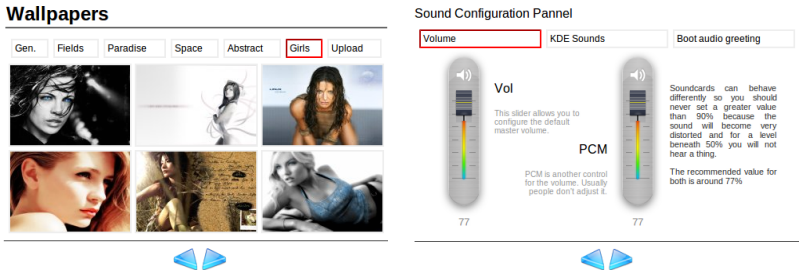
1. Vytvoření adresářové struktury LiveCD
2. Připojení Virtual File System (VFS), ten zahrnuje složky: /sys /proc /dev /dev/shm /dev/pts /var/run
3. Nainstalování základních balíčků
4. Nainstalování modulů squashfs a aufs
5. Nainstalování kernelu a vytvoření initramfs pomocí mkinitcpio s minimálně aktivovanými 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.

7.2.2 NimbleX

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



Obr. 1. Webové rozhraní s nastavením NimbleX, v levo výběr míry nastavení, v pravo nastavení programů v sekci office



Obr. 2. Webové rozhraní s nastavením NimbleX, v levo výběr wallpaperu, v pravo nastavení zvuku

7.2.3 Linux-Live

Linux-Live je sada scriptů, která není závislá na distribuci, ale lze ji použít na jekémkoliv linuxu. Prakticky stačí stáhnou scripty ze stránek linux-live a zkompilovat si kernel s podporou aufs a squashfs nebo si stáhnout již zkompilovaný kernel ze stránek linux-live. Scripty už pak samy vytvoří Live system, který je možné nabootovat z CD nebo USB.

7.2.4 Další možnosti tvorby LiveCD

Linux-live je nejčastější způsobem vytvoření LiveCD. Spousta distribucí má, ale vlastní nástroje některé jako: The Ubuntu Customisation Kit (UCK) nebo Reconstructor, které již byly popsány (viz. kapitola 5.4.2/s26). 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/s28) 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 několik jednoduchými kroky lze docílit vytvoření LiveCD na bázi Archlinuxu.

II. PROJEKTOVÁ ČÁST

8 LFS BY BASH SCRIPTS

Účelem LFS by bash scripts je sestavení distribuce LFS za pomocí jednoduchých a přehledných scriptů 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.

8.1 Struktura a použití

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

8.1.1 Adesářová struktura

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

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

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 potřébné pro uživatelský profil LFS

linker.1.test/linker.1.test64 dva soubory, které slouží pro test funkcnosti při oddělení toolchainu.

makeall hlavní spouštěcí script, který spuští vše potřebné až do konce kapitoly 5.

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

README.markdown README soubor ve snačkovacím jazyce markdown

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

version-check.sh Oficiální script 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**

functions script se všemi funkcemi

ver script s poly, které obsahují informace o balíčcích

l10n-EN už téměř nepoužívaný script, který měl sloužit pro lokalizaci do jiných jazyků, byl zavrhnut, protože znamenal velkém množství porměnných

8.1.2 Důležité proměnné

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

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 připojem 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 scripty

Každý script 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 scriptu 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 8.1.3/s43) 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 8.1.3/s44) (http://… ftp://…)

MD5= ${glibc[5]} do proměnné se ukládá kontrolní součet staženého souboru pro test správnosti stažení, využívaný funkcí check (viz. kapitola 8.1.3/s44) (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.

8.1.3 functions

script 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 scriptu.

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 paramatru 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 testovány 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 paramatrem je název souboru patche, který je větsinou 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 scripty.

```
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 scriptu, dokud uživatel nezmáčkne klavesu enter. Podobnou funkcí je pak fuknce waitUserAl(), která pozastaví scripty 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 mno chyb při kompilaci.

8.2 Návod pro sestavení

Sestavení je díky scriptům vcelku jednoduchý proces. Podle návodu v LFS uživatel vytvoří prázdný diskovy 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 musejí 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 dobré stáhnout a překopírovat scripty LFS by bash scripts. Nejjednoduším způsobem je stáhnutí scriptů pomocí verzovacího systému git.

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

a ještě překopírovat soubory užvatele 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
```

po těchto krocích je už třeba jen přihlásit se jako uživatel lfs a spustit script makeall

```
su - lfs
cd $LFS
sh ./scripts/makeall
```

script 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 script prepareChap6, který vytvoří potřebné složky a virtuální souborové systému.

```
# sh ./scripts/prepareChap6
```

nyní je třeba se chrootnout do systému, tedy se přihlásit do námi vytvořného toolchainu

```
chroot "$LFS" /tools/bin/env -i \
    HOME=/root TERM="$TERM" PS1='\u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin:/tools/bin \
    /tools/bin/bash --login +h
```

a spustit script chrootMakeall

```
../scripts/chrootMakeall
```

ZÁVĚR

text

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.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

LFS	Linux From Scratch
OS	Operační systém
HW	Hardware
SW	Sogtware
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 Criteries
CCRA	Common Criteria Recognition Arrangement
EAL	Evaluation Assurance Levels
SElinux	Security-Enhanced Linux
LSM	Linux Security Modules
IPC	interprocess communication
SSH	Secure Shell

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I. NÁZEV PŘÍLOHY