TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Metody distribuce a nasazení zákaznického software v prostředí OS Linux

The distribution and deployment methods in the Linux OS environment

Bakalářská práce

Studijní program: B2646 – Informační technologie

Studijní obor: 1802R007 – Informační technologie

Autor práce: **Patrik Jíra**

Vedoucí práce: Ing. Lenka Kosková Třísková, Ph.D

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Metody distribuce a nasazení zákaznického software v prostředí OS Linux

Jméno a příjmení: **Patrik Jíra**
Osobní číslo: M15000025
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2018/2019**

Zásady pro vypracování:

1. Seznamte se s možnostmi úprav a sestavení distribucí OS Linux.
2. Seznamte se s licencemi OSS a možnostmi jejich užití v průmyslové praxi.
3. Navrhněte postup sestavení vlastní distribuce OS Linux včetně doplňkových balíků software s ohledem na zvolenou licenční politiku.
4. Sestavte distribuci OS Linux dle požadavků, hotové řešení dokumentujte.

Rozsah grafických prací:	dle potřeby
Rozsah pracovní zprávy:	30 – 40 stran
Forma zpracování práce:	tištěná/elektronická

Seznam odborné literatury:

[1] SALVADOR, Otavio a Daiane ANGOLINI. Embedded Linux development with Yocto project. Birmingham: Packt Publishing, 2014. ISBN 978-1-78328-233-3.
[2] Laurent Andrew M. St. Understanding Open Source and Free Software Licensing. O'Reilly Media, Inc., 2004. ISBN 0596005814.
[3] Love, R. Linux Kernel Development. Addison-Wesley Professional, 2010. ISBN 0672329468.

Vedoucí práce:	Ing. Lenka Kosková Třísková, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	18. října 2018
Předpokládaný termín odevzdání:	30. dubna 2019

L. S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci 18. října 2018

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jeho využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí.

Datum:

Podpis:

Poděkování

Rád bych poděkoval vedoucí mé bakalářské práce Ing. Lence Koskové Třískové, Ph.D. za poskytnutí odborných rad, ochotu a vstřícný přístup během celé práce na dané problematice.

Abstrakt

Cílem této bakalářské práce je vytvořit a zdokumentovat vlastní distribuci operačního systému založeného na Linuxu. K dosažení tohoto cíle je nutné prozkoumat vhodné možnosti pro vytvoření uživatelsky upravené verze operačního systému. Následně z daných možností vybrat tu nejvhodnější a tímto způsobem vytvořit dle předem daných specifikací distribuci operačního systému.

Klíčová slova

Distribuce, Linux, Operační systém, Yocto

Abstract

The goal of presented bachelor thesis is to create and document our own distribution of operating system based on Linux. To reach this goal, you need to explore the appropriate options for creating a customized version of the operating system. Subsequently, from the given options to choose the most suitable and in this manner create the distribution of the operating system according to the predetermined specifications.

Keywords

Distribution, Linux, Operating system, Yocto

Seznam obrázků

Seznam grafů

Seznam tabulek

Seznam zkratek

AFL	Academic Free License
AGPL	Affero General Public License
BSD	Berkeley Software Distribution
GPL	General Public License
ICU	International Components for Unicode
LGPL	Lesser General Public License
MPL	Mozilla Public License
OFL	Open Font License
PSF	Python Software Foundation License

Obsah

1 Úvod

V oblasti průmyslové automatizace a informačních technologií se neustále zvyšují nároky na řídící systémy. Je kladen velký důraz na rychlost, uživatelskou přívětivost, možnosti nasazovaných aplikací, ale také jsou kladeny stále větší nároky na rychlost implementace systémů. Vytváření aplikačních řešení pro jednotlivé koncové uživatele sebou nese velký problém a to sice ten, že každý koncový uživatel má jiné nároky a požadavky na výslednou aplikaci. Tento fakt velice ztěžuje vytvoření takové aplikace, která by natolik pokryla možnosti trhu, aby nebylo nutné každé řešení začínat jako kompletně nový projekt.

Na popud těchto skutečností vznikla myšlenka vytvořit novou Linuxovou distribuci, který bude figurovat jako jádro uceleného systému dílčích aplikací. Součástí tohoto systému bude několik služeb, které pokrývají velkou část požadavků koncových uživatelů. Uživatel si tyto služby sám zvolí, podle svých potřeb a nebude tudíž nutné vytvářet pro každého uživatele samostatné řešení. Veškerá konfigurace OS je přístupná pouze z konzole, do které nemá koncový uživatel přístup.

Pro vytvoření upraveného operačního systému je nutné seznámení se s nástroji, které toto nastavení umožňují. Při zpracování této práce byl kladen důraz na několik faktorů, které zajistí bezproblémovou implementaci aplikací do systému jak po technické, tak právní stránce. Jedním z těchto faktorů je zabezpečení systému před nepovolenými zásahy ze strany koncových uživatelů, ale také zabezpečení přístupu k zdrojovým kódům aplikací, které do systému budou zakomponovány.

2 Licence

Před započetím jakékoliv programátorské činnosti je nutné si uvědomit, že prostředí, ve kterém se budu při této práci pohybovat, se skládá z mnoha balíčků a každý je vytvořený pod jinou licencí. Proto je nutné u každého ze softwarových balíčků zhodnotit použitou licenci, zdali je vůbec právně možné ho zakomponovat do výsledné práce.

Softwarová licence je právní nástroj, kterým se řídí použití nebo redistribuce softwaru. Podle autorského zákona je veškerý software chráněn autorskými právy, a to jak ve zdrojovém kódu, tak ve formách kódů objektů. Jedinou výjimkou je software ve veřejné doméně. Typická softwarová licence uděluje nabyvateli licence, koncovému uživateli povolení používat jednu nebo více kopií softwaru způsobem, kdy by takové použití jinak mohlo představovat porušení autorských práv výhradními právy vlastníka softwaru podle autorského práva. [1]

Dle autorského zákona je umožněno vlastníkovi určité kopie softwaru používat software s počítačem, i když použití softwaru s počítačem vyžaduje zhotovení některých kopií nebo úprav. Proto je vlastník kopie počítačového softwaru oprávněn používat tuto kopii softwaru. Pokud je tedy koncový uživatel softwaru vlastníkem příslušné kopie, může koncový uživatel legálně používat software bez licence vydavatele softwaru. [2]

Vlastnictví digitálních výrobků, jako jsou softwarové aplikace a videohry, je zpochybňováno licencovanými, smlouvami EULA digitálních distributorů, kterým je např. společnost Steam. V Evropské unii Evropský soudní dvůr rozhodl, že držitel autorských práv nemůže bránit dalšímu prodeji digitálně prodávaného softwaru, v souladu s pravidlem vyčerpání autorských práv při prvním prodeji. [2]

Bezplatné licence poskytují uživateli licence práva, podobná právům vlastníka. Nabyvatel licence může například kopírovat, upravovat a distribuovat díla za předpokladu, že je získána bezplatná licence. [3]

S proprietárním softwarem si původní vlastník autorských práv ponechává vlastnictví. Poskytnutím licence, která není vždy právně závazná, vlastník autorských práv pronajímatelům licence pronajímá materiál chráněný autorskými právy. [3]

Následující kapitoly obsahují popis vybraných licencí, které jsou nejčastěji používané ve výchozí distribuci. Dále se zde nacházejí licence, které sice nejsou natolik početně zastoupeny v původní distribuci, ale jsou pro danou problematiku zajímavé svými licenčními podmínkami. Popis licencí začíná tabulkou, ve které jsou nejprve licence porovnány.

	Používání	Kopírování	Úprava	Distribuce	Sublicence	Doplňující informace
MIT	ANO	ANO	ANO	ANO	ANO	Jediná nutnost při distribuci je přiložit kopii znění licence a jméno autora
GPL	ANO	ANO	ANO	Ano, ale pod stejnou licencí	NE	U služeb běžících na serveru není nutné poskytovat pozměněné zdrojové kódy
AGPL	ANO	ANO	ANO	Ano, ale pod stejnou licencí	NE	I u služeb běžících na serveru je nutné poskytnout zdrojový kód
LGPL	ANO	ANO	ANO	ANO	ANO	Knihovny lze kombinovat s jakýmkoliv SW a pouze použitá knihovna zůstává pod licencí LGPL
Freetype	ANO	ANO	ANO	ANO	ANO	V dokumentaci je nutné uvést části použité z originální aplikace a uvést u těchto částí jejich původní licenci
BSD	ANO	ANO	ANO	Ano, ale pod stejnou licencí	ANO	Všechny části použitého zdrojového kódu musí obsahovat text licence BSD. Nicméně nevyžaduje distribuci zdrojového kódu
Apache	ANO	ANO	Ano, ale musejí být popsané	ANO	ANO	Není nutné zveřejňovat zdrojový kód, ale je nutné vypsat všechny změny provedené v originální verzi
AFL	ANO	ANO	ANO	ANO	ANO	Je nutné poskytnout čitelnou kopii zdrojového kódu originální kopie a výsledného produktu
MPL	ANO	ANO	ANO	ANO	ANO	Kód je možné kombinovat s jinými licencemi, ale převzatá část musí stále splňovat podmínky MPL
PSF	ANO	ANO	ANO	ANO	NE	Umožňuje modifikované verze distribuovat bez zdrojového kódu
OFL	ANO	ANO	ANO	ANO	NE	Výsledek úpravy musí splňovat licenci OFL
ICU	ANO	ANO	ANO	Ano, ale pod stejnou licencí	NE	Licenční ICU podmínky musejí být vypsány v každé kopii

Tabulka 1: Porovnání licencí

2.1 MIT

Používání kopie softwaru nesoucí licenci MIT[1] je bezplatně dovoleno jakékoli osobě, která získá kopii daného softwaru včetně dokumentačních souborů k němu náležející. Se softwarem je dovoleno zacházet bez omezení práv na používání, kopírování, upravování, slučování s dalším softwarem, zveřejňování, distribuování, poskytování sublicencí nebo prodávání kopií softwaru za podmínky toho, že každá kopie nebo část použitého softwaru bude obsahovat doslovné znění licence MIT-style a jméno autora původní verze aplikace. [4]

2.2 GPL

GPL[2] je široce používaná licence svobodného softwaru, která zaručuje koncovým uživatelům svobodu spouštět, studovat, sdílet a upravovat software. Licenci původně napsal Richard Stallman z Nadace pro svobodný software (FSF) pro projekt GNU a uděluje příjemcům počítačového programu práva Definice svobodného softwaru. GPL je licence copyleft, což znamená, že odvozené dílo může být distribuováno pouze za stejných licenčních podmínek. [5]

[1] Celý text licence MIT je k nalezení zde: https://wiki.pirati.cz/kci/mit_licence

[2] Celý text licence GPL je k nalezení zde: https://www.gnu.org/licenses/gpl.html

GPLv1[3] byla vydána, aby zabránila dvěma hlavním metodám, jakými vývojáři softwaru omezovaly svobody, které definují svobodný software.

Prvním problémem původní verze je to, že nebyl přímo specifikován způsob, jakým musí být software publikovaný. Vývojáři této mezery využívali a software publikovali pouze jako binární soubory, které se sice dali spustit, ale nebylo možné je modifikovat nebo přečíst. Aby se tomuto problému zamezilo, tak se v GPLv1 uvádí, že kopírování a distribuce kopií nebo jejich částí programu musí také zpřístupnit zdrojový kód čitelný člověkem za stejných licenčních podmínek. [5]

Druhým problémem původní verze bylo to, že vývojáři mohli přidávat omezení buď do licence, nebo kombinací softwaru pod licencí GPL se softwarem s jinou licencí, které podmínky byli přísnější a omezovala tak práva svobodného software. Od verze GPLv1 je tedy možné kombinovat software pouze tak, aby byl výsledek distribuován pod licenčními podmínkami GPLv1. Software distribuovaný pod podmínkami GPLv1 by tedy mohl být kombinován se softwarem za podmínek, které jsou tolerantnější, protože by to nezměnilo podmínky, za kterých by mohl být celý distribuován. [5]

GPLv2[4] a její klauzule "Liberty of Death" byla hlavní změnou v této verzi. Tato sekce říká, že nabyvatelé licence mohou distribuovat dílo, na které se vztahuje licence GPL, pouze pokud mohou splnit všechny podmínky této licence. Povinnosti vyplývající z licence nesmí být přerušeny z důvodu konfliktních událostí. Účelem tohoto ustanovení je odradit jakoukoli stranu od uplatnění nároku na porušení patentu, který by narušil svobodu uživatelů na základě licence. [6]

[3] Celý text licence GPLv1 je k nalezení zde: https://www.gnu.org/licenses/gpl.html

[4] Celý text licence GPLv2 je k nalezení zde: https://www.gnu.org/licenses/old-licenses/gpl-2.0.txt

GPLv3[5] byla vytvořena aby změnila definici softwarových patentům, svobodné softwarové licenční kompatibilitě, definici "zdrojového kódu" a hardwarovým omezením softwarové modifikace. Dále upravuje způsob, jakým jsou řešeny porušování licencí, a způsob, jakým držitel autorských práv může udělit další oprávnění. [7]

GPLv3 zlepšila kompatibilitu s několika licencemi pro open source software, jako je například Apache License, verze 2.0 a GNU Affero General Public License, s nimiž GPLv2 nemohl být kombinován. Software GPLv3 však lze kombinovat a sdílet pouze se softwarem GPLv2, pokud používaná licence GPLv2 s volitelnou klauzulí „nebo novější“ a software byl upgradován na GPLv3. [7]

2.3 AGPL

GNU Affero General Public License (AGPL[6]) je bezplatná licence copyleft, kterou vydala Nadace Free Software Foundation v listopadu 2007 a která je založena na GNU General Public License, verzi 3 a Všeobecné veřejné licenci Affero. [8]

Původně navržena jako způsob, jak učinit software „svobodnějším“ než GPL tím, že nutí dostupnost softwaru založeného na AGPL. AGPL je skutečně užitečná pro obchod, protože zatímco umožňuje vývojářům a konkurentům přístup ke zdrojovému kódu, v zásadě znemožňuje novým účastníkům konkurovat v dané problematice s původním vývojářem komerčně, protože musí zveřejňovat své změny, zatímco původní vývojář, může stále nabízet komerční licence. [8]

[5] Celý text licence GPLv3 je k nalezení zde: https://www.gnu.org/licenses/gpl.html

[6] Celý text licence AGPL je k nalezení zde: https://www.gnu.org/licenses/agpl-3.0.html

2.3.1 Kompatibilita s GPL

Licence GNU AGPLv3 a GPLv3 obsahují klauzule, které společně dosahují formy vzájemné kompatibility obou licencí. Tyto klauzule výslovně povolují "přenášení" díla vytvořeného spojováním kódu licencovaným pod jednou licencí proti kódu licencovanému na základě jiné licence, a to navzdory tomu, že licence jinak neumožňují opětovné znovuzískání licencí za podmínek, které jsou mezi sebou. Tímto způsobem je copyleft každé licence uvolněn, aby umožnil distribuci takových kombinací. [9]

2.4 Freetype

Licence Freetype[7] je inspirována licencemi BSD, artistic a IJG z nichž všechny podporují volné využití software v komerčních a volně dostupných produktech. Uděluje na celém světě, bez licenčních poplatků, trvalé právo a povolení k užívání, spouštění, zobrazení, kopírování, vytváření odvozených děl, distribuce a sublicencování projektu FreeType Project ve formě (zdrojového i objektovém kódu) a jejich odvozené dílo pro jakýkoli účel za splnění následující podmínky: *Redistribuce zdrojového kódu musí zachovat soubor se celým nezměněným textem licence. Všechny dodatky, odstranění nebo změny původní doklady musí být uvedeny v dokumentaci. Dále musí být zachovány nezměněné údaje o autorských právech v původních souborech a ve všech kopiích zdrojových souborů.* [10]

2.5 LGPL

Licence LGPL[8] umožňuje vývojářům a společnostem používat a integrovat softwarové komponenty uvolněné pod LGPL do vlastního (i proprietárního) softwaru, aniž by to vyžadovaly podmínky silné licence copyleft pro uvolnění zdrojového kódu vlastních komponent. Každý vývojář, který modifikuje komponentu licencovanou pod LGPL, je však povinen zpřístupnit upravenou verzi pod stejnou licencí LGPL. Pro proprietární software, kód pod LGPL je obvykle používán ve formě sdílené knihovny, tak že tam je jasné oddělení

[7] Celý text licence Freetype je k nalezení zde: https://git.savannah.gnu.org/cgit/freetype/freetype2.git/tree/docs/FTL.TXT

[8] Celý text licence AGPL je k nalezení zde: https://opensource.org/licenses/lgpl-3.0.html

mezi proprietárními a LGPL komponenty. LGPL je primárně používán pro softwarové knihovny, i když je používán i v některých samostatných aplikacích. [11]

2.5.1 Rozdíl mezi LGPL a GPL

Hlavní rozdíl mezi GPL a LGPL spočívá v tom, že umožňuje, aby byla práce spojena s programem, který není LGPL, bez ohledu na to, zda se jedná o svobodný software nebo proprietární software. Program, který není LGPL, pak může být distribuován za jakýchkoliv podmínek, pokud to není odvozené dílo. Pokud se jedná o odvozené dílo, pak musí podmínky programu umožnit "modifikaci pro vlastní použití zákazníka a reverzní inženýrství pro ladění takových úprav." Samostatný spustitelný soubor, který dynamicky navazuje na knihovnu prostřednictvím .so, .dll nebo podobného média, je obecně přijímán jako není odvozené dílo, jak je definováno LGPL. [12]

2.6 BSD

Licence BSD[9] jsou skupinou povolených licencí svobodného softwaru, které ukládají minimální omezení pro používání a distribuci softwaru. Tato skutečnost je v rozporu s copyleft licencemi, které mají požadavky na sdílení. Původní verze byla od doby jejího vzniku revidována a její potomci jsou označováni jako modifikované licence BSD. [13]

BSD je jak licence, tak třída licencí. Upravená BSD licence jsou velmi podobné licenci původně používané pro BSD verzi Unixu. Licence BSD je jednoduchá licence, která vyžaduje pouze to, aby všechny části kódu uchovávaly oznámení o licenci BSD, pokud jsou redistribuovány ve formátu zdrojového kódu, nebo pokud byly distribuovány v binárním formátu. Licence BSD nevyžaduje, aby byl zdrojový kód distribuován. [13]

[9] Celý text licence k nalezení zde: https://wiki.pirati.cz/kci/bsd_licence

2.6.14-clause license

Původní BSD licence obsahovala klauzuli, která nebyla nalezena v pozdějších licencích, známou jako "reklamní doložka". Tato klauzule se nakonec stala kontroverzní, jak to vyžadovali autoři všech prací pocházejících z BSD licence. Podmínkou pro distribuci aplikace pod licencí BSD s touto klauzulí je zahrnout potvrzení originálního zdroje ve všech reklamních materiálech. [13]

2.6.23-clause licence

Tato verze byla prověřena jako Open source licence "Licence BSD". Nadace Free Software Foundation, která odkazuje na licenci jako „modifikovanou BSD licenci“, uvádí, že je kompatibilní s GNU GPL. Při vytváření softwaru pod touto licencí je nutné přesně specifikovat o jakou verzi licence se jedná, aby se zamezilo záměně s původní verzí licence. [13]

BSD 3-clause umožňuje neomezenou redistribuci pro jakýkoliv účel, pokud jsou dodržena autorská práva a odmítnutí záruky. Licence také obsahuje doložku omezující použití jmen přispěvatelů k potvrzení odvozeného díla bez zvláštního povolení. [13]

2.6.32-clause licence

Zjednodušená verze se začala používat, především pro své použití ve FreeBSD. Primární rozdíl mezi ním a novou licencí BSD spočívá v tom, že vynechává klauzuli o neschválení. Nadace Free Software Foundation, která odkazuje na licenci jako licenci FreeBSD, uvádí, že je kompatibilní s GNU GPL. Kromě toho FSF vybízí uživatele, aby byli specifičtí, když odkazují na licenci podle jména. [13]

2.7 ICU

Open-source projekt knihoven C / C ++ a Java pro podporu Unicode, internacionalizaci softwaru a globalizaci softwaru. ICU[10] je široce přenosný do mnoha operačních systémů a prostředí. Poskytuje aplikacím stejné výsledky na všech platformách a mezi softwarem C, C ++ a Java. Projekt ICU je sponzorován, podporován a využíván společností IBM a mnoha dalšími společnostmi. [14]

"Oprávnění se tímto bezplatně uděluje jakékoli osobě, která získá kopii tohoto softwaru a souvisejících dokumentačních souborů (dále jen „software"), aby se se softwarem zacházela bez omezení, včetně, bez omezení, práv na používání, kopírování, úpravy, sloučování , zveřejňovat, distribuovat a / nebo prodávat kopie Softwaru a umožnit tak osobám, kterým je Software poskytován, za předpokladu, že výše uvedená oznámení o autorských právech a toto oznámení o oprávnění jsou uvedeny ve všech kopiích Softwaru a že výše uvedená oznámení o autorských právech a toto oznámení o povolení se zobrazí v podpůrné dokumentaci."

2.8 Apache

Licence Apache[11] je povolená licence svobodného softwaru napsaná Apache Software Foundation (ASF). Licence Apache License, verze 2.0 vyžaduje zachování autorských práv a prohlášení o vyloučení odpovědnosti. Licence, stejně jako ostatní licence na svobodný software, umožňuje uživateli softwaru svobodně používat software k jakémukoli účelu, distribuovat jej, upravovat a distribuovat upravené verze softwaru v souladu s podmínkami licence, bez obav pro licenční poplatky. [15]

Verze 1.1 Apache License byla schválena v roce 2000. Odvozené produkty již nejsou povinny zahrnout do svých reklamních materiálů přiřazení, pouze v jejich dokumentaci. [16]

[10] Celý text licence ICU je k nalezení zde: https://spdx.org/licenses/ICU

[11] Celý text licence Apache je k nalezení zde: https://www.apache.org/licenses/LICENSE-2.0

Verze 2.0 byla schválena v roce 2004. Mezi uvedené cíle licence patřilo zjednodušení používání licence pro jiné než ASF projekty, zlepšení kompatibility se softwarem založeným na GPL, umožňující zahrnutí licence do referencí. [16]

2.9 Zlib

Licence zlib[12] byla schválena nadací Free Software Foundation (FSF) jako licence na svobodný software a Open Source Initiative (OSI) jako open source licence. Je kompatibilní s GNU General Public License. [17]

Distribuce upravené verze softwaru podléhá následujícím omezením:

1. Autorství původního softwaru nesmí být zkresleno, změněné verze nesmí být zkresleny jako původní software a oznámení o licenci nesmí být odstraněno ze zdrojových distribucí.
2. Licence nevyžaduje, aby byl zdrojový kód zpřístupněn při distribuci binárního kódu. [17]

2.10 AFL

Akademická bezplatná licence (AFL[13]) je povolená licence svobodného softwaru. Licence uděluje obdobná práva jako licence BSD, MIT, UoI / NCSA a Apache licence, které umožňují, aby byl software proprietární. [18]

2.11 MPL

Mozilla Public License (MPL[14]) je bezplatná a otevřená softwarová licence vyvinutá a spravovaná nadací Mozilla Foundation. Je to slabá copyleftová licence, která je charakterizována jako střední cesta mezi povolenými licencemi svobodného softwaru a licencí GPL, která usiluje o rovnováhu mezi zájmy autorizovaných a open source vývojářů. [19]

---

[12] Celý text licence zlib je k nalezení zde: http://www.libpng.org/pub/png/src/libpng-LICENSE.txt

[13] Celý text licence AFL je k nalezení zde: https://opensource.org/licenses/AFL-3.0

[14] Celý text licence k nalezení zde: https://spdx.org/licenses/MPL-1.1.html

2.12 OFL

Licence SIL Open Font (OFL[15]) je bezplatná a open source licence určená pro fonty SIL International pro použití s mnoha jejich fonty Unicode, včetně Gentium Plus, Charis SIL a Andika. [20]

2.13 PSF

Licence Python Software Foundation License (PSF[16]) je licencovaná licence BSD ve stylu svobodného softwaru, která je kompatibilní s GNU General Public License (GPL). Jeho primární použití je pro distribuci Python projektového softwaru. Na rozdíl od GPL licence Python není licencí copyleft a umožňuje modifikované verze distribuovat bez zdrojového kódu. [21]

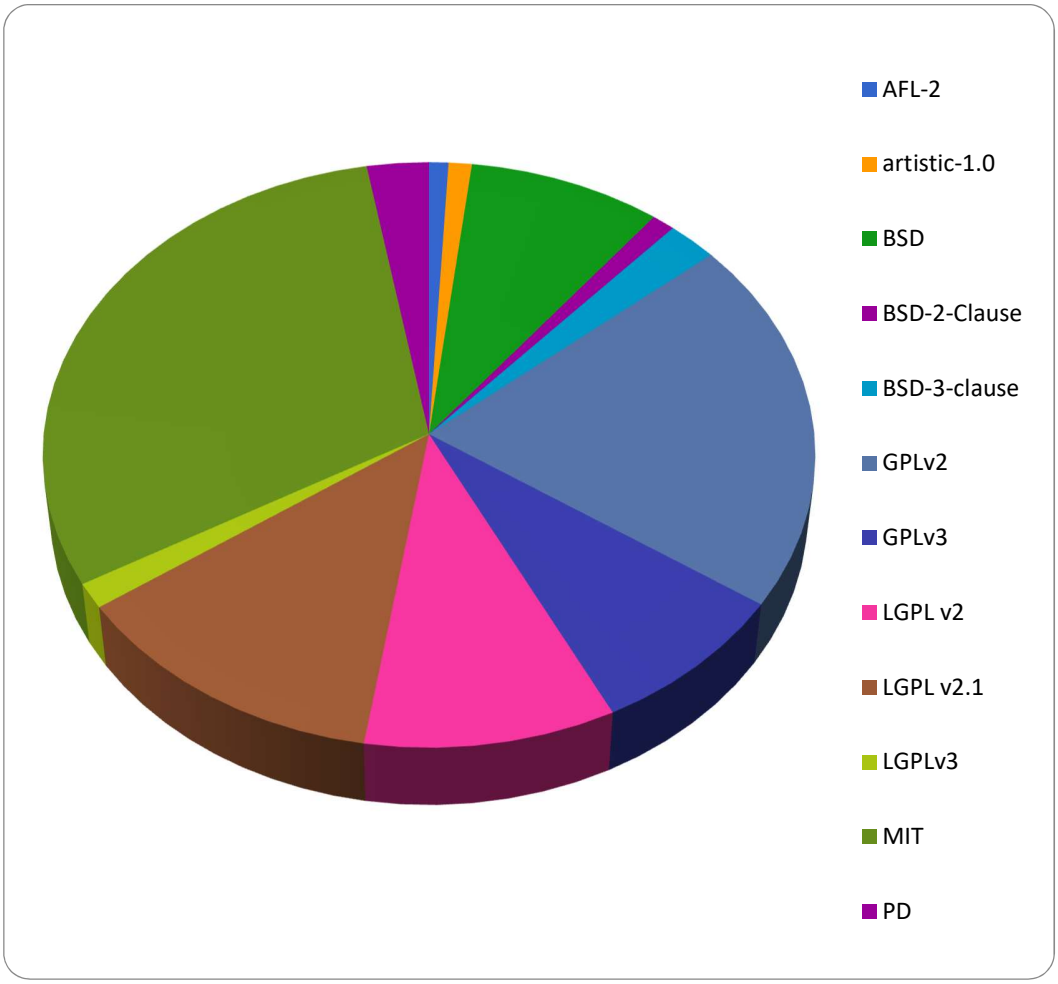
2.14 Využití více licencí

Některé balíčky použité v operačním systému využívají více licencí, což je způsobeno tím, že se daný produkt skládá z více části produktů, které jsou licencované pod různými licencemi, tím pádem výsledek tohoto sloučení musí nést licence všech použitých balíčků. Nicméně existuje také možnost tohoto sloučení, kde si můžeme vybrat z nabídky licencí, kterou vlastně použijeme.

Na následujím grafu jsou zobrazené četnosti licencí balíčku v Linuxové distribuci Ubuntu, z níž tato vychází. Graf obsahuje pouze licence s výskytem větším než 1%.

[15] Celý text licence k nalzení zde: https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=ofl

[16] Celý text licence k nalezení zde: https://docs.python.org/3/license.html



Graf 1: Četnosti licencí v Ubuntu

2.15 Shrnutí licenční problematiky

O úspěchu tvořené aplikace rozhoduje několik faktorů a jeden z nich je dobře zvolená licenční politika. Pro výsledný produkt, bude aplikace, která je předmětem této bakalářské práce sloužit jako jádro. To je také důvod, proč je nutné zvolit vhodné licence již v rané fázi tohoto vývoje a to sice u samotného jádra. Pro potřeby zabezpečení je důležité zvolit takové licence tak, aby ochránili aplikace, které budou nadstavbou zmiňovaného jádra.

Celá kapitola Licence je proto věnována licencím, které již používají jednotlivé balíčky operačních systému Ubuntu a jejich stručnému popisu, popřípadě porovnání daných licencí. Vývoj tohoto operačního systému není započat od samého začátku, ale vychází z již existujícího operačního systémů, kde je zachováno samotné jádro, funkce a aplikace na

tomto systému postavené. Je také nutné podle zvolené licenční politiky odstranit jednotlivé aplikace a balíčky vytvořené aplikace tak, aby s těmito licencemi nekolidovali a byli možné je dále využívat pro své účely.

Jako vhodné licence pro použití ve výsledném operačním systému se pro úpravu aplikační vrstvy ukázaly všechny licence, které dovolují používání, kopírování, úpravu, distribuci, možnost vytvoření sublicence, ale co je hlavní tak licence, které nevyžadují zveřejnění zdrojového kódu. Tyto podmínky splňují licence MIT, v případě zprovoznění aplikace na serveru licence GPL, LGPL, BSD, PSF. Aplikace a funkce ponechané z výchozího operačního systému, které nesplňují zvolenou licenční politiku zůstanou nezměněné a bude s nimi pracováno v rámci dané licence.

3 Možnosti sestavení distribuce

V této kapitole jsou popsané vybrané prostředky, s jejichž pomocí je možné sestavit Linuxovou distribuci podle požadavků uživatele. Téměř každá z větších distribucí Linuxu (podle počtu přispěvovatelů a uživatelů) vyvinula svou verzi systému pro sestavení dané distribuce podle potřeb uživatele. Celá práce je zaměřená na dvě ze čtyř nejrozšířenějších a mě nejbližších linuxových distribucí a to sice openSUSE a Ubuntu.

3.1 Ubuntu minimal CD

Minimální ISO obraz funguje tak, že stahuje balíčky z online archivů v době instalace namísto jejich poskytnutí na samotném instalačním médiu. Stahování balíčků v době instalace snižuje velikost obrazu iso souboru na přibližně 40 MB v závislosti na architektuře a poskytuje pouze balíčky potřebné pro instalaci. Úspora času stahování dosažená použitím mini iso může být významná, protože jsou staženy pouze aktuální balíčky, takže není nutné aktualizovat balíčky ihned po instalaci. [22]

3.2 openSUSE

Služba pro sestavení obsahuje projekty, kde každý projekt obsahuje zdroje potřebné pro sestavení jednoho nebo více balíčků (tj. RPM / DEB / atd.). Tyto zdroje zahrnují zdrojové archivy, soubory verzí, atd. Výstupem projektu je jeden nebo více repozitářů. Úložiště je svazek RPM organizovaný v hierarchii adresářů spolu s některými soubory indexu / meta-dat, které usnadňují nástrojům jako je zypper vyhledávání a řešení závislostí. Repozitáře a výstupy projektu odpovídají různým verzím operačního systému, jako je openSUSE 11.2 atd. [23]

3.3 Yocto project

Projekt Yocto je open source projekt, který pomáhá vývojářům vytvářet vlastní Linuxové distribuce, jenž jsou určeny pro vestavěné produkty bez ohledu na hardwarovou architekturu. Projekt Yocto poskytuje flexibilní sadu nástrojů a vývojové prostředí, které umožňuje vývojářům vestavěných zařízení po celém světě spolupracovat prostřednictvím

sdílených technologií, softwarových balíčků, konfigurací a osvědčených postupů používaných k vytvoření těchto přizpůsobených obrazů Linuxu. [24]

Projekt je standardem, pokud jde o vývoj vestavěných softwarových balíčků. Umožňuje přizpůsobení softwaru a možnost vývoje pro více hardwarových platforem, stejně jako softwarové balíčky, které lze udržovat a měnit. [24]

3.4 Shrnutí možností sestavení

Hned z několika níže uvedených důvodu byl pro vytvoření linuxové distribuce zvolen jako nejvhodnější project Yocto. Samostatné balíčky sice povětšinou nabízejí stejné možnosti, nicméně jen pro jednu danou distribuci. Je velice pravděpodobné, že v budoucnu dojde ke změně výchozí linuxové distribuce, respektive chceme tuto možnost nechat otevřenou a project yocto toto poměrně jednoduše umožňuje. Samotné sestavování operačního systému je u projectu yocto také jedinečné. Ostatní výše zmíněné využívají aplikační nástroje s uživatelským rozhraním, ve kterém je možné pouze vybrat, které balíčky jsou v distribuci požadovány a které ne. Project yocto využívá pro sestavení nástroj zvaný bitbake a důmyslný systém vrstev a tříd pomocí kterého je možné do výsledné distribuce přidat uživatelsky vytvořené aplikace, které samotná distribuce vůbec nenabízí. Další nespornou výhodou je možnost přizpůsobit výslednou distribuci pro různé procesory jako například ARM. Což nechává otevřenou možnost nasazení softwaru na vestavěných (embedded) zařízení.

4 Yocto project

V Následující kapitole jsou podrobněji vysvětleny jednotlivé dílčí body Yocto projectu, což pomůže lépe pochopit možnosti práce s projektem Yocto.

4.1 Struktura projektu

Struktura Yocto je členitá, proto se v následujících odstavcích pokusím popsat nejdůležitější části projektu, což pomůže přiblížit a pochopit základní strukturu projektu Yocto. Jednotlivé části struktury na sebe navazují a při sestavování vlastní distribuce je nutné dodržovat tuto strukturu. Celý výsledný operační systém je "poskládán" z těchto dílčích částí.

Soubory append jsou soubory, které upravují nastavení určitého receptu. Tyto soubory jsou označovány příponou .bbappend. Při sestavování systém očekává, že každý připojený soubor bude mít odpovídající soubor receptů (.bb). Soubor s příponou a odpovídající soubor receptu musí navíc používat stejný kořenový název souboru. Názvy souborů se mohou lišit pouze v použité příponě typu souboru. (example.bb a example.bbappend). [25]

Recept je soubor instrukcí pro sestavování balíčků. Recept popisuje, kde lze získat zdrojový kód, kterou verzi použijete, jak konfigurovat zdroj a jak jej kompilovat. Recepty také popisují závislosti pro knihovny nebo jiné recepty. [25]

BitBake je základní složkou projektu Yocto a používá ho sestavovací systém OpenEmbedded pro vytváření obrazů. Jedná se o automatický generátor úloh, který umožňuje efektivně a paralelně spouštět úlohy shell a Python při práci v rámci komplexních omezení závislosti mezi úkoly. Lze pomocí něj ovládat celé sestavení vlastního obrazu. Zpracovává jednotlivé recepty, vytváří sestavení a výsledný obraz operačního systému. [25]

OpenEmbedded-Core (OE-Core) je společná vrstva metadat používaných systémy odvozenými z OpenEmbedded, která zahrnuje projekt Yocto. Projekt Yocto a OpenEmbedded Project udržují OpenEmbedded Core. Metadata OE-Core jsou přístupné v úložištích zdrojů projektu Yocto.

Openembedded-core vzniklo ve spolupráci yocto projekt a společnosti OpenEmbedded čímž vzniklo kvalitní a stabilní jádro. [25]

Konfigurační soubory obsahují globální definice proměnných, uživatelem definované proměnné a informace o konfiguraci hardwaru. Tyto soubory sdělují Open-Embedded systému, co překládat a co dát do obrazu, aby podpořily konkrétní platformu. [25]

Extensible Software Development Kit (eSDK) je vlastní SDK pro vývojáře aplikací. Tato eSDK umožňuje vývojářům začlenit své knihovny a programové změny zpět do obrazu, aby jejich kód byl k dispozici dalším vývojářům aplikací. Informace o eSDK naleznete v příručce Vývoj aplikací aplikací Yocto a v příručce Extensible Software Development Kit (eSDK). [25]

Metadata jsou klíčovým prvkem projektu Yocto. Ta se používají k vytvoření distribuce Linuxu a jsou obsažena v souborech, které systém OpenEmbedded sestavuje při vytváření obrazu. Metadata obecně zahrnují recepty, konfigurační soubory a další informace, které se týkají samotných instrukcí sestavení, a také data, která se používají k řízení toho, co se staví. Metadata zahrnují i příkazy a data, která se používají k označení toho, jaké verze softwaru jsou používány, odkud jsou získány změny nebo dodatky k samotnému softwaru (opravy nebo pomocné soubory), které se používají k opravě chyb nebo přizpůsobení softwaru pro použití v softwaru. [25]

Vrstva je sbírka souvisejících receptů. Vrstvy umožňují konsolidovat související metadata a přizpůsobit tak sestavení. Izolují také informace použité při vytváření více architektur a jsou hierarchické ve své schopnosti přepsat předchozí specifikace. Můžete zahrnout libovolný počet dostupných vrstev z projektu Yocto a přizpůsobit sestavení přidáním vrstev za nimi. [25]

4.2 Vrstvy

Model vrstev projektu Yocto je vývojový model pro linuxové systémy označované jako embedded nebo IoT, která odlišuje projekt Yocto od jiných jednoduchých sestavovacích systémů. Vrstvy jsou repositáře, které obsahují související sady instrukcí, které sdělují systému OpenEmbedded, co má dělat. Vrstvy mohou spolupracovat a opakovaně se používat. [25]

Vrstvy mohou kdykoliv obsahovat změny předchozích pokynů nebo nastavení. Tato schopnost přepsání umožní přizpůsobit dříve sestavené vrstvy tak, aby vyhovovaly požadavkům.

Vrstvy se používají pro logické oddělení jednotlivých částí výsledné aplikace. Například BSP, GUI, konfiguraci distro, middleware nebo aplikační vrstvy. Pokud by všechny části aplikace byly uvedeny v jedné vrstvě, tak by pozdější úpravy a případné použití jen dané části programu bylo komplikované. Naproti tomu izolace informací do vrstev pomáhá zjednodušit budoucí úpravy a opětovné použití. Čím více modulární jsou metadata, tím je snazší provádět v budoucnu změny. [25]

Každé vrstvě je přiřazena priorita:

1. OpenEmbedded Core Metadata;
2. Yocto-specific layer Metadata;
3. Hardware-specific Metadata;
4. UI-specific Layer;
5. Commercial Layer;
6. Developer-specific Layer;

Tato priorita určuje pouze pořadí, v jakém se budou jednotlivé vrstvy překládat (1-6). [25]

4.3 Poky

Poky je referenční distribuce Yocto Project. Obsahuje OpenEmbedded sestavovací systém (BitBake a OpenEmbedded Core), stejně jako sadu metadat, která umožní vybudovat vlastní distro. Poky neobsahuje binární soubory, je to příklad, jak vytvořit vlastní distribuci Linuxu ze zdroje. [26]

1. Funkční distro na základní úrovni slouží k ilustraci, jak přizpůsobit distribuci;
2. Prostředek, pomocí kterého se testují komponenty projektu Yocto Project (tj. Poky se používá k ověření projektu Yocto);
3. Prostředník, pomocí kterého si můžete stáhnout projekt Yocto; [26]

Poky nabízí open source nástroj založený na Linuxu, X11, Matchbox, GTK+, Pimlico, Clutter a dalších mobilních technologií založených na GNOME. Poskytuje stabilní verzi OpenEmbedded, na které je možné spolehlivě sestavovat distribuce a následně je distribuovat. Poky podporuje velkou množinu podporovaných hardwarových zařízení x86, ARM, MIPS a PowerPC. [26]

Poky je určen pro tvorbu platforem, který generuje obrazy souborového systému založené na open source softwaru, jako je například server Kdrive X, Matchbox, sada nástrojů GTK + a systém sběrnic dat D-Bus. Zatímco obrazy pro mnoho druhů zařízení mohou být generovány, standardní příklad stroje cílí QEMU plně simulovanému systému (x86, ARM, MIPS a PowerPC) a skutečné referenční desky pro každou z těchto architektur. Schopnost instrukce zavádět se uvnitř emulátoru QEMU ho činí obzvláště vhodným jako testovací platforma pro vývoj vestavěného softwaru. [26]

Důležitou součástí integrovanou do Poky je Sato, prostředí uživatelského rozhraní založeného na platformě GNOME Mobile. Je navržen tak, aby dobře fungoval s obrazovkami, které používají velmi vysoké hodnoty DPI a mají omezené velikosti, například ty, které se často nacházejí na smartphonech a PDA. Vzhledem k tomu, že Sato je kódován pro rychlost a efektivitu, funguje hladce na ručním i jiném vestavěném hardwaru. [26]

4.4 Sestavení obrazu

4.4.1 Požadavky na sestavení

Hostitelský systém s minimálně 50 GB volného místa na disku, na kterém běží podporovaná distribuce Linuxu. Pokud hostitelský systém podporuje více jader a vláken, je možné upravit konfiguraci sestavení projektu Yocto Project tak, aby výrazně zkrátil čas potřebný k vytvoření obrazů (ISO). [27]

Nainstalované balíčky potřebné pro práci s yocto projektem. Požadavky na balíčky se liší podle sestavované distribuce.

Stroj, na kterém bude probíhat nastavení nové distribuce musí obsahovat specifický systém, ze kterého bude nová distribuce vycházet, což je v případě této práce Ubuntu.

Podporované distribuce: Ubuntu, openSUSE, CentOS, Debian [27]

Potřebné balíčky: Git 1.8.3.1+, tar 1.27+, Python 3.4.0+. Tyto balíčky jsou nutné pro všechny distribuce. [27]

4.4.2 Příklady vzorových konfiguracích

build-appliance-image - příklad virtuálního stroje, který obsahuje všechny potřebné části ke spuštění sestavení pomocí systému sestavení. Obraz je možné spustit pomocí VMware Player nebo VMware Workstation.

core-image-base: Obraz pouze s konzolí, který podporuje hardware cílového zařízení.

core-image-clutter: Obraz s podporou nástroje Open GL Toolkit Clutter, který umožňuje vývoj bohatých a animovaných grafických uživatelských rozhraní.

core-image-full-cmdline: Pouze konzolový obraz s více plnohodnotnými funkcemi systému Linux.

core-image-lsb: Obraz, který odpovídá specifikaci Linux Standard Base (LSB). Tento obraz vyžaduje distribuční konfiguraci, která umožňuje soulad s LSB.

core-image-minimal: Malý obraz, který je schopen umožnit spuštění zařízení.

core-image-minimal-dev: Obraz, který je vhodný pro vývoj práce s hostem. Obraz obsahuje knihovny, které lze použít v prostředí vývoje hostitele.

core-image-minimal-initramfs: Obraz, u kterého filesystém využívá minimum paměti RAM jako součást kernelu, který umožňuje systému efektivněji najít první program „init“.

core-image-minimal-mtdutils který má podporu pro minimální MTD nástroje, jenž umožňují uživateli komunikovat se subsystémem MTD v jádře, což dovoluje provádět operace na flash zařízeních.

core-image-rt: Obraz, který obsahuje sadu nástrojů, které jsou vhodné pro testování v reálném čase.

core-image-rt-sdk: obraz, který obsahuje všechno v cross-toolchainu. Obraz také obsahuje knihovny, které tvoří kompletní samostatný SDK a je vhodný pro vývoj pomocí cíle.

core-image-sato: Obraz s podporou Sato, mobilním prostředím a vizuálním stylem, který funguje dobře s mobilními zařízeními. Obraz podporuje X11 s motivem Sato a aplikacemi, jako je terminál, editor, správce souborů, přehrávač médií a tak dále.

core-image-sato-dev: Obraz obsahuje knihovny potřebné pro tvorbu aplikací na samotném zařízení, nástroje pro testování a profilování a symboly ladění.

core-image-testmaster: Obraz "master" určený pro automatické testování. Poskytuje obraz, který je nasazen do samostatného oddílu, takže je možné jej zavést a použít k nasazení druhého obrazu, který má být testován.

core-image-weston: Velmi základní obraz Waylandu s terminálem. Tento obraz poskytuje knihovnu protokolů Wayland.

core-image-x11: Velmi základní X11 obraz s terminálem. [25]

4.5 QEMU

Zatímco systémy virtualizace, jako jsou Xen a KVM, spoléhají na rozdělení zdrojů hostitelského počítače pro různé hostované operační systémy, QEMU emuluje jiný systém. Emulace zahrnuje software, který funguje jako rozhraní mezi hostujícím systémem a skutečným hardwarem. Z hlediska hostitele emulátor představuje všechna rozhraní hardwaru, která by normálně viděla na emulovaném systému. Namísto kódu prováděného hostem, který je přímo prováděn na základním hardwaru, emulátor interpretuje instrukce pro emulovaný systém do instrukcí, které může vykonávat základní hardware. To znamená, že prostřednictvím procesu emulace lze software pro jeden systém spustit v emulátoru běžícím na nekompatibilním hardwaru. [28]

Hostitelský systém musí být silnější než hostovaný systém, který je emulován, aby host mohl vykonávat instrukce v přijatelné rychlosti. Vzhledem k těmto nákladům se emulace běžně používá k tomu, aby umožňovala používání softwaru, pro který je těžké získat nebo koupit originální hardware. Stejně jako u hosta na kompatibilním hardwaru s hostitelem, tak virtualizace prostřednictvím hypervizoru může poskytnout mnohem lepší výkon. [28]

QEMU je schopna emulovat řadu různých hardwarových platforem, včetně x86 PC, PowerPC platforem, jako jsou PowerMac, systémy založené na ARM, jako jsou Nokia N800 a Palm Tungsten a také systémy SUN SPARC. Kromě základního hardwaru těchto systémů poskytuje QEMU také emulaci řady různých přídavných modulů, jako jsou grafické karty, zvukové karty, síťová zařízení, paměťová zařízení a řadiče, sériová / paralelní / USB zařízení a paměťová zařízení. To znamená, že v mnoha případech je možné emulovat a provozovat kompletní a plně funkční počítače ke spuštění původního softwaru. [28]

Díky své schopnosti emulovat řadu periferií je QEMU často používán ve spojení s hypervizory XEN a KVM, aby poskytoval virtuálním strojům schopnost interakce s emulovanými periférierni, což poskytuje funkce, jako je vytváření sítí a zvuku, které nemusí být jinak být k dispozici prostřednictvím čisté virtualizace. V těchto scénářích QEMU jednoduše poskytuje emulovaná hardwarová rozhraní pro tato periferní zařízení, což ponechává řízení provádění kódu hostujícího systému na samotném hypervisoru. [28]

Zajímavou další funkcí, která je k dispozici prostřednictvím QEMU, je schopnost provádět emulaci uživatelského režimu. To znamená, že QEMU může umožnit operačnímu systému spouštět software, který mohl být kompilován pro jinou hardwarovou platformu, než emulovat úplný systém včetně operačního systému. Příkladem této simulace je spuštění binárního softwaru pro 32bitový systém ARM Linux na 64bitovém systému x86 Linux. Jedná se o omezenou funkci, i když QEMU může běžet na systémech Linux, Windows, UNIX a MacOS. Tato funkce je dostupná pouze pro systémy Linux a některé systémy UNIX. [28]

4.6 Doplňující aplikace

Aplikace, které fungují na tomto systému jako služby, tyto aplikace si koncový uživatel sám zvolí a v průběhu používání je kdykoliv možné tyto volby aplikací měnit (přidávat/ubírat). Jedná se převážně o aplikace týkající se průmyslové automatizace. Aplikace na sběr dat z průmyslových počítačů (PLC). Dále aplikace, které budou určovat prediktivní údržbu. VPN pro vzdálené připojení. Condition monitoring, který se bude starat o správnou funkčnost celé linky a plynulost výroby.

5 Návrh sestavení

Kapitola, která je zaměřena na popis postupu při sestavení distribuce a její úpravě podle požadavků zadavatele.

5.1 Požadavky zadavatele

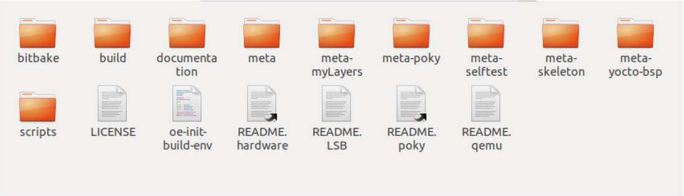
1. Možnosti šifrování disku.
2. Zamezení konfigurace operačního systému koncovým uživatelem.
3. GUI – určené pouze pro ovládání aplikací, které na systému budou fungovat.
4. Důležité je zamezit koncovému uživateli přístup do konzole, zdrojovým kódům aplikací.
5. Vytvoření uživatelských účtů v instalačním souboru.
6. Přidání aplikace do instalačního souboru.

5.2 Prvotní sestavení obrazu

Pro urychlení sestavení je vhodnější využít některého balíčku z možností Yocto projectu, který sestavuje pouze konzolový systém.

1. Vytvoření lokální kopie poky (git clone http://git.yoctoproject.org/git/poky);
2. Následně přejdeme do vytvořené kopie poky (cd poky);
3. Vytvoření svého projektu (git checkout -b VasNazev origin/"VasNazev");
4. Vytvoření složky nutné pro sestavení (source oe-init-build-env). V této složce se následně nacházejí informace nutné pro sestavení systému (vrstvy, které se v sestavení mají/nemají použít, konfigurační soubory, atd...);
5. Následně příkaz bitbake -k core-image-full-cmdline spustí sestavení operačního systému;
6. Po dokončení sestavení je k dispozici jedna ze základních verzí námi zvoleného operačního systému, kterou můžeme dále upravovat (Doba tohoto sestavení je závislá na použitém stroji a hlavně na použité vzorové konfiguraci);

7. Podle základního nastavení se operační systém vygeneruje pro virtuální stroj QEMU, ve kterém je možné s vygenerovaným systémem pracovat;
8. Příkazem runqemu qemux86 se tento virtuální stroj spustí; [27]



Obrázek 1: Vygenerovaná struktura

Po dokončení zmíněného postupu budeme mít k dispozici nezměněnou strukturu operačního systému, která je znázorněna na obrázku výše.

5.2.1 build

Obsahuje konfigurační soubory nutné k nastavení distribuce.

1. Seznam vrstev, které se mají sestavit;
2. Seznam vrstev, které se mají nainstalovat;
3. Informace o stroji, na který je výsledná distribuce cílená;
4. Inicializaci proměnných, které se při sestavení využívají;
5. Postahované balíčky, které jsou připravené pro zakomponování;
6. Již sestavené distribuce;

5.2.2 bitbake

Obsahuje konfigurační soubory nástroje na nastavení bitbake.

5.2.3 meta-poky

Vrstva, která obsahuje soubory s konfigurací systému pro sestastení. Dále tato vrstva obsahuje nástroje busybox, psplash a tiny-init.

BusyBox je softwarová sada, která poskytuje několik unixových nástrojů v jediném spustitelném souboru. Běží v různých prostředích POSIX, jako je Linux a Android, i když mnoho nástrojů, které poskytuje, je navrženo pro práci s rozhraními, které poskytuje jádro Linuxu. Vytvořen pro embedded operační systémy s velmi omezenými zdroji.

Tento spustitelný soubor nahrazuje základní funkce více než 300 běžných příkazů. Vydáván jako svobodný software za podmínek GNU. [29]

PSplash je uživatelská grafická úvodní obrazovka pro hlavně embedded Linux zařízení podporující 16bpp nebo 32bpp framebuffer. Má několik závislostí, podporuje základní obrazy a text. Vizuální vzhled lze konfigurovat pomocí základních změn zdrojových kódů. Součástí je také příkazový nástroj klienta pro odesílání informací do psplash, jako jsou informace o postupu spouštění. [30]

Tiny-init obsahuje základní systém pro inicializaci. Normálně je proces inicializace zajišťován operačním systémem. Balíček tiny-init je užitečný v prostředích, které nemají nativní proces init. [31]

meta-selftest obsahuje konfigurační soubory vývojových nástrojů určených pro překládání spustitelných souborů (python, c, ...).

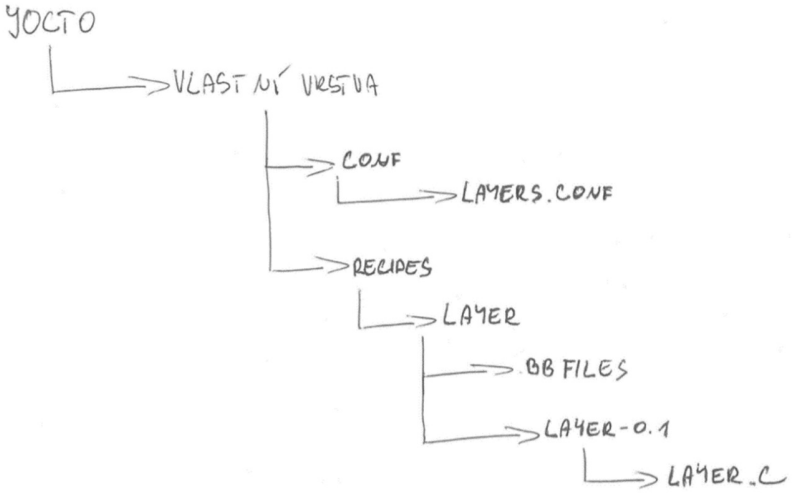
meta-skeleton obsahuje vzorové konfigurace pro veškeré možné úpravy, jaké jsou v projectu yocto dovolené.

meta-yocto-bsp obsahuje konfigurační soubory jádra a grafického uživatelského rozhraní.

5.3 Úprava vrstev

Při práci s vrstvami je nezbytné dodržovat strukturu, která je v projectu yocto určena. Každá vrstva, popřípadě skupina vrstev potřebuje konfigurační soubory, ve kterých jsou informace o dané vrstvě/skupině vrstev. Struktura zobrazená na snímku níže je základní struktura, kterou lze vytvořit ručně přidáním daných složek nebo pomocí generátoru bitbake a to sice pomocí příkazu bitbake-layers create-layer. Vygenerovaná struktura obsahuje i základní bb soubory a konfigurační soubory, nicméně tyto soubory obsahují pouze obecné informace, které je potřeba upravit/doplnit.

U testování nové vrstvy není nutné pokaždé překládat celý systém a mnohem rychlejší cestou je překládat pouze danou aplikaci, což je možné příkazem bitbake název_vrstvy.



Obrázek 2: Struktura vlastní vrstvy

Ve složce conf se nachází soubory ve kterých je uložena konfigurace. V této konfiguraci se upravují proměnné, které jsou důležité pro sestavení a pracuje s nimi bitbake při generování struktury výsledné distribuce.

Ukázka konfiguračního souboru pro nastavení vrstev:

```
BBPATH .= ":${LAYERDIR}"
BBFILES += "${LAYERDIR}/recipes-*/*/*.bb \
        ${LAYERDIR}/recipes-*/*/*.bbappend"
BBFILE_COLLECTIONS += "myLayers"
BBFILE_PATTERN_myLayers = "^${LAYERDIR}/"
BBFILE_PRIORITY_myLayers = "5"
FILESEXTRAPATHS_prepend := "${THISDIR}/Ownlayer-0.1:"
```

BBPATH: Používaná nástrojem pro nastavení BitBake k vyhledání souborů třídy (.bbclass) a konfiguračních souborů (.conf). Tato proměnná je analogická s proměnnou PATH. Pokud je BitBake spuštěn z adresáře mimo adresář sestavení, tak je nutné aby byla proměnná BBPATH nastavena tak, aby odkazovala na adresář sestavení. [32]

BBFILES uvádí seznam souborů, který určuje, které soubory má BitBake "hledat" při sestavení. [32]

BBFILE_COLLECTIONS uvádí názvy konfigurovaných vrstev. Tato proměnná slouží k nalezení dalších proměnných BBFILE. Každá vrstva obvykle připojuje svůj název k této proměnné ve svém souboru conf / layer.conf. [32]

BBFILE_PATTERM_myLayers proměnná, která expanduje tak, aby odpovídala souborům z BBFILES v určité vrstvě. Tato proměnná se používá v souboru conf / layer.conf a musí být doplněna názvem konkrétní vrstvy. [32]

BBFILE_PRIORITY_myLayers přiřadí prioritu pro soubory receptů v každé vrstvě.Tato proměnná je důležitá hlavně v situacích, kdy se stejný recept objevuje ve více než jedné vrstvě. Nastavení této proměnné umožní upřednostnit vrstvu proti jiným vrstvám, které obsahují stejný recept. Přednost stanovená touto proměnnou je bez ohledu na verzi receptu. Vrstva, která má recept s vyšší hodnotou verze receptu, ale pro kterou je BBFILE_PRIORITY nastavena na nižší prioritu, má stále nižší prioritu. [32]

FILESEXTRAPATHS_prepend proměnná, která upřesňuje cestu k jednotlivým receptům. [32]

5.3.1 Nastavení uživatelů

Přidání uživatelů do samotné instalace operačního systému funguje v projectu yocto jako každá jiná úprava a to sice v podobně vrstvy. Přidání uživatelů, skupin a úprava zabezpečení roota je možné udělat v jedné vrstvě.

Ukázka receptu pro úpravu uživatelů:

```
SUMMARY = "User add"
SECTION = "examples"
LICENSE = "CLOSED"
S = "${WORKDIR}"
PACKAGES =+ "${PN}-custom"
inherit useradd
EXTRA_USERS_PARAMS = "usermod -P root root;"
USERADD_PACKAGES = "${PN}"
USERADD_PARAM_${PN} = "-u 1200 -d /home/custom -r -s /bin/bash
-P 123 custom"
do_install_append () {
     install -d -m 755 ${D}${datadir}/custom
     chown -R custom ${D}${datadir}/custom}
FILES_${PN} = "${datadir}/*"
```

PACKAGES seznam balíčků, které recept vytvoří. [33]

inherit useradd tato část zdrojového kódu udává, že kód, který následuje dědí z třídy useradd. Třída useradd podporuje přidání uživatelů nebo skupin. Pokud jsou použity například balíčky obsahující systémové služby, které by měly být spouštěny pod vlastním uživatelem nebo skupinou, je možné tuto třídu použít k vytvoření uživatele nebo skupiny. Recept meta-skeleton / recipes-skeleton / useradd / useradd-example.bb ve zdrojovém adresáři poskytuje jednoduchý příklad, který ukazuje, jak přidat tři uživatele a skupiny do dvou balíčků. [33]

EXTRA_USERS_PARAMS pomocí této proměnné je možné pracovat s třídou extrausers. Zdědění třídy extrausers globálně nebo z obrazového receptu umožňuje provádět další operace uživatelů a skupin. Třída extrauser umožňuje použití další konfigurace uživatele a skupiny na úrovni obrazu. [33]

USERADD_PACKAGES při zdědění třídy useradd tato proměnná určuje jednotlivé balíčky v rámci receptu, které vyžadují přidání uživatelů nebo skupin. Tato proměnná musí být nastavenat, pokud recept zdědí třídu useradd. Například v ukázkovém kódu je pomocí této proměnné přidán uživatel custom s heslem 123. [33]

5.3.2Přidání vlastní aplikace

Přidání aplikace do výsledné distribuce funguje podobně, jako úprava nastavení uživatelů. Nicméně v tomto případě je nutné dát si pozor na zadanou licenci. S ohledem na zadanou licenci se poté odvíjí možnost práce na přidané aplikaci ostatními vývojáři.

Ukázka receptu přidání vrstvy:

```
DESTRIPTION = "Simple layer"
SECTION = "examples"
LICENSE = "MIT"
LIC_FILES_CHKSUM="file://${
COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"
PR = "r0"
SRC_URI = "file://ownlayer.c"
S = "${WORKDIR}"
do_compile(){
              ${CC} ${CFLAGS} ${LDFLAGS} ${WORKDIR}/ownlayer.c
              -o ownlayer
}
do_install() {
               install -m 0755 -d ${D}${bindir} ${D}${do-
               cdir}/ownlayer
          install -m 0111 ${S}/ownlayer ${D}${bindir}
}
```

LICENSE seznam licencí pro daný recept.

LIC_FILES_CHKSUM kontrolní součty licenčního textu ve zdrojovém kódu receptu. Proměnná, pomocí které se sledují změny v licenčním textu souborů zdrojového kódu. Pokud dojde ke změně licenčního textu, dojde k selhání sestavení, což dává možnost zpětně kontrolovat jakoukoli změnu licence. Tato proměnná musí být definována pro všechny recepty (pokud není LICENSE nastavena na "CLOSED"). [33]

PR udává verzi receptu. Výchozí hodnota této proměnné je "r0". [33]

SRC_URI je seznam zdrojových souborů (lokálních nebo vzdálených). Tato proměnná říká systému OpenEmbedded sestavení, které soubory a jak má zpracovávat. Pokud například soubor receptu nebo soubor append potřebuje pouze načíst soubor receptu nebo přidaného souboru použije jeden SRC_URI vstup. Na druhou stranu, pokud soubor receptu nebo přílohy vyžaduje použití dvou souborů a zahrnutí vlastního souboru, soubor receptu nebo append soubor by zahrnoval několik instancí této proměnné. [33]

6 Zabezpečení

Zabezpečení je v tomto případě nutné řešit ze dvou hlavních důvodů. Zamezení přístupu k nastavení operační systému a dílčím aplikacím. A zabezpečení zdrojových kódů přidaných aplikací. Do jisté míry je možné zabezpečit instalační soubory aplikací správným nastavení oprávnění. Nicméně tento způsob funguje jen v případě zamezení přístupu k rootu.

6.1 Přístup do konzole

Přístupu do konzole koncovému uživateli je možné zamezit úplně, nicméně v tomto případě je plně dostačující zamezit přístupu k právům roota. V případě uložení instalačních souborů fyzicky na disku stačí už jen u daných souborů nastavit práva přístupu tak, aby měl obyčejný uživatel pouze možnost daný soubor spustit.

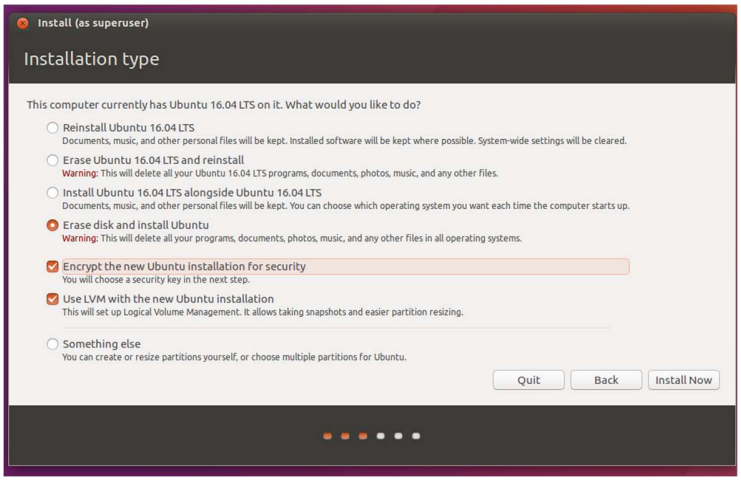
6.2 Zabezpečení souborů na disku

Existují dva způsoby, jakými lze šifrovat disk. Šifrování celého disku nebo jen určitého oddílu. V našem případě je určitě vhodnější způsob šifrovat pouze jeden oddíl disku a to pouze pokud budou uchovávány zdrojové kódy přímo na disku. V případě uchování citlivých dat na disku poté zašifrovat části disku, na kterých se tyto informace nacházejí. Nicméně mnohem bezpečnější je využití formy vzdáleného privátního úložiště na principu git. Git v projektu yocto dokáže plně nahradit zdrojový kód uložený na disku a v případě využití gitu není nutné se dodatečným šifrováním zabývat.

6.2.1 Šifrování celého disku

Zašifrovat celý disk je možné hned při zavádění operačního systému na disk. Při zavádění operačního systému se nejprve objeví obrazovka, na které je možné vybrat zdali chceme vyzkoušet ubuntu bez jakýchkoliv změn na disku, což znamená, že je operační systém zaveden přímo z média na kterém se nachází. Nicméně nastavení šifrování celého disku je možné pouze v případě instalování operačního systému na lokální disk.

Při pokračování se zvolením instalace následuje část, ve které je vybrána možnost šifrování celého disku. Tento postup je znázorněn na snímku níže. V průběhu šifrování disku je smazán celý obsah disku.



Obrázek 3:Nastavení šifrování celého disku

V následujícím kroku instalace je požadováno zvolení hesla, které je používáno k odemknutí šifrovaných dat při každém spuštění počítače. Následně pokračuje instalace jako bez šifrování, to znamená zvolení hesla pro uživatele a pro administrátora, časového pásma, rozložení klávesnice, atd..

Při každém spuštění operačního systéme je nutné zadat heslo, které bylo zvoleno pro zašifrování dat.

6.2.2Šifrování oddílu

Pokud nepotřebuje šifrovat celý disk, ale pouze určitá citlivá data je vhodné tyto data oddělit na samostatný oddíl disku a šifrovat pouze tuto část disku. V našem případě celý tento postup obstarává vrstva, kterou je nutné patřičně nakonfigurovat. V podstatě se jedná pouze o sadu příkazů, které se postupně vykonají.

Nejprve je nutné do systému nainstalovat balíček, pomocí kterého lze docílit požadovaného šifrování. Proto je první zadaný příkaz sudo apt-get install cryptsetup-bin. Jakmile se balíček crypsetup nainstaluje jsou splněny všechny požadavky pro šifrování. Následně je nutné vybrat příslušný oddíl. Zobrazit oddíly na disku lze příkazem sudo lsblk.

V našem případě bude součástí vrstvy pouze příkaz s již vybraným oddílem sudo cryptsetup luksFormat/dev/název_oddílu. Následuje systémová hláška, které upozorňuje, že celý oddíl disku bude zformátován a zdali jsme zálohovali data.

Po naformátování celého oddílu je nutné vyplnit přístupové fráze pro přístup k šifrované oblasti, bez které není možné manipulovat se soubory v daném oddílu. V tomto okamžiku se šifrovaný uzamknutý oddíl zobrazí ve spouštěči. Při pokusu přistoupit k zašifrované části disky je vždy vyžadována fráze, která byla zvolena jako klíč.

Oddíl disku je zašifrovaný, ale ještě není nijak nastavený. Proto je nutné pomocí nástroje disk utility tento oddíl najít a naformátovat ho s požadovaným souborovým systémem a s požadovaným názvem.

7 Závěr

Tato bakalářská práce měla několik základních cílů. Prvním z nich bylo seznámit se s dostupnými možnostmi úprav a sestavení distribucí OS Linux. Těchto možností existuje několik, proto jsem se rozhodl v této práci zaměřit na tři způsoby, kterými lze sestavit vlastní verzi Linuxové distribuce. Díky rozsáhlým dokumentacím jsem byl schopen každý způsob otestovat a vybrat z nich ten nejvhodnější. Jednotlivé možnosti jsou zdokumentované a následně jsou uvedeny důvody, díky kterým je vybrán nejvhodnější způsob pro pokračování v kapitolách následujících. V práci se dále primárně zabývám pouze projektem yocto, což je ze tří testovaných způsob s nejrozsáhlejšími možnostmi pro úpravu distribuce.

Druhým důležitým bodem bylo seznámení se licencemi a licenční politikou, které se v Linuxovém a opensource prostředí využívají a jejich možnosti využití v praxi. Ubuntu, ze kterého má práce vychází obsahuje přibližně 450 softwarových balíčků, ve kterých je použita většina známých licencí pro otevřený software. Informace obsažené v této práci týkající se licencí se proto odvíjejí od těchto balíčků. Samotný vývoj aplikací, které by mohli fungovat na systému založeném na distribuci Linux se meze ohledně licenční politiky nekladou. Nicméně je nutné je dodržovat a uvědomit si, z jakého důvodu jsou tyto aplikace vyvíjené. Pokud je to pro generování zisku, tak bych určitě doporučil striktně copyleftové licence, převážně poté takové, které nevyžadují sdílení zdrojového kódu. Samozřejmě, pokud vyvíjená aplikace vychází již z jiného balíčku, je nutné dodržovat licenční podmínky originálu.

Třetí a poslední čtvrtý bod této bakalářské se již odvíjejí pouze od projectu Yocto a jeho struktury. Otestoval jsem postup, pomocí kterého je možné vytvořit lokální kopii kterékoliv distribuce, kterou Yocto project nabízí. Tento postup jsem následně pospal v dokumentaci. Samozřejmostí je popsání jednotlivých součástí projektu, které je nutné znát při práci v prostředí Yocto projectu.

Následuje zdokumentovaný postup sestavení distribuce podle požadavků. V této části jsem se zabýval možnostmi zabezpečení, uživatelským přístup a přidání vlastní aplikace. Podle informací v návodu, který je poskytnut v této dokumentaci je možné sestavit

vlastní distribuci. Ve které je možné zabezpečit přístup k nastavení systému a tím zajistit bezpečný chod výsledné aplikace pomocí omezení přístupu koncového uživatele do konzole, zabezpečení dat, které se nacházejí na disku pomocí dodatečného šifrování, přidání vlastní aplikace do výsledné distribuce.

Dle závěrů a postupů obsažených v této práci je možné sestavit vlastní distribuci Linuxu pro embedded, IoT i PC zařízení. Zde použitá metodika je jedním z mnoha postupů, kterými lze dosáhnout požadovaných cílů. Práce se zabývá poměrně rozsáhlou problematiku licenční politiky, která je nedílnou a nutnou součástí celé této problematiky.

Citovaná literatura

[1] M. Rouse, „techtarget,“ Červen 2014. [Online]. Available: https://searchcio.techtarget.com/definition/software-license. [Přístup získán 15 Březen 2019].

[2] P. wikipedie, „wikipedia,“ 21 Duben 2019. [Online]. Available: https://en.wikipedia.org/wiki/Software_license. [Přístup získán 23 Duben 2019].

[3] Techopedia, „techopedia,“ [Online]. Available: https://www.techopedia.com/definition/2558/software-licensing. [Přístup získán 25 Březen 2019].

[4] Techopedia, „techopedia,“ [Online]. Available: https://www.techopedia.com/definition/3287/mit-license. [Přístup získán 15 Březen 2019].

[5] L.-C. (. Tai, „free-soft,“ 4 Červenec 2001. [Online]. Available: http://www.free-soft.org/gpl_history/. [Přístup získán 15 Březen 2019].

[6] F. S. Foundation, „gnu,“ Červen 1991. [Online]. Available: https://www.gnu.org/licenses/old-licenses/gpl-2.0.txt. [Přístup získán 18 Březen 2019].

[7] gnu, „gnu,“ 18 Listopad 2016. [Online]. Available: https://www.gnu.org/licenses/gpl.html. [Přístup získán 18 Březen 2019].

[8] F. G. Marand, „quora,“ 7 Březen 2018. [Online]. Available: https://www.quora.com/What-is-AGPL-license. [Přístup získán 18 Březen 2019].

[9] apsillers, „stackexchange,“ 29 Srpen 2017. [Online]. Available: https://opensource.stackexchange.com/questions/5909/is-every-license-that-is-gplv3-compatible-also-agplv3-compatible. [Přístup získán 15 Březen 2019].

[10] D. Turner, R. Wilhelm a W. Lemberg, „savannah,“ 27 Leden 2006. [Online]. Available: https://git.savannah.gnu.org/cgit/freetype/freetype2.git/tree/docs/FTL.TXT. [Přístup získán 20 Březen 2019].

[11] Luk, „abclinux,“ 1 Listopad 2005. [Online]. Available: http://www.abclinuxu.cz/slovnik/gnu-lgpl. [Přístup získán 2 Duben 2019].

[12] fluendo, „fleundo,“ 31 Srpen 2018. [Online]. Available: https://fluendo.com/en/blog/post/gpl-vs-lgpl/. [Přístup získán 2 Duben 2019].

[13] P. wikipedie, „wikipedia,“ 22 Březen 2019. [Online]. Available: https://en.wikipedia.org/wiki/BSD_licenses. [Přístup získán 10 Březen 2019].

[14] P. wikipedie, „wikipedia,“ 16 Únor 2015. [Online]. Available: https://cs.wikipedia.org/wiki/International_Components_for_Unicode. [Přístup získán 5 Duben 2019].

[15] M. Rouse, „techtarget,“ Srpen 2013. [Online]. Available: https://whatis.techtarget.com/definition/Apache-License. [Přístup získán 22 Březen 2019].

[16] P. wikipedie, „wikipedia,“ 9 Duben 2019. [Online]. Available: https://en.wikipedia.org/wiki/Apache_License. [Přístup získán 15 Duben 2019].

[17] P. wikipedie, „wikipedia,“ 31 Leden 2019. [Online]. Available: https://en.wikipedia.org/wiki/Zlib_License. [Přístup získán 12 Duben 2019].

[18] L. Rosen, „opensource,“ 2005. [Online]. Available: https://opensource.org/licenses/AFL-3.0. [Přístup získán 10 Duben 2019].

[19] „mozilla,“ [Online]. Available: https://www.mozilla.org/en-US/MPL/. [Přístup získán 10 Duben 2019].

[20] theleagueof, „github,“ 23 Srpen 2010. [Online]. Available: https://theleagueof.github.io/licenses/ofl-faq.html. [Přístup získán 11 Duben 2019].

[21] 19 Duben 2019. [Online]. Available: https://docs.python.org/3/license.html. [Přístup získán 21 Duben 2019].

[22] quiverc, „ubuntu,“ 7 Říjen 2018. [Online]. Available: https://help.ubuntu.com/community/Installation/MinimalCD. [Přístup získán 20 Duben 2019].

[23] openSUSE, „opensuse,“ 3 Březen 2019. [Online]. Available: https://en.opensuse.org/openSUSE:Build_Service_Tutorial. [Přístup získán 4 Březen 2019].

[24] yoctoproject, „yoctoproject,“ 2018. [Online]. Available: https://www.yoctoproject.org/about/. [Přístup získán 15 Duben 2019].

[25] yoctoproject, „yoctoproject,“ Únor 2019. [Online]. Available: https://www.yoctoproject.org/docs/current/mega-manual/mega-manual.html. [Přístup získán 20 Březen 2019].

[26] yoctoproject, „yoctoproject,“ Prosinec 2011. [Online]. Available: https://www.yoctoproject.org/docs/1.0.2/poky-ref-manual/poky-ref-manual.html#intro-welcome. [Přístup získán 16 Duben 2019].

[27] yoctoproject, „yoctoproject,“ 2015. [Online]. Available: https://www.yoctoproject.org/docs/1.8/yocto-project-qs/yocto-project-qs.html. [Přístup získán 20 Duben 2019].

[28] H. Smith, „vps,“ [Online]. Available: https://www.vps.net/blog/what-is-qemu/. [Přístup získán 24 Duben 2019].

[29] openembedded, „busybox,“ Duben 2019. [Online]. Available: https://layers.openembedded.org/layerindex/recipe/5249/. [Přístup získán 24 Duben 2019].

[30] openembedded, „openembedded,“ [Online]. Available: https://layers.openembedded.org/layerindex/recipe/587/. [Přístup získán 15 Duben 2019].

[31] RKrahl, „github,“ 30 Červenec 2018. [Online]. Available: https://github.com/RKrahl/tiny-init. [Přístup získán 21 Březen 2019].

[32] yoctoproject, „yoctoproject,“ Únor 2019. [Online]. Available: https://www.yoctoproject.org/docs/2.6.1/ref-manual/ref-manual.html#structure-core-meta-poky. [Přístup získán 9 Duben 2019].

[33] yoctoproject, „yoctoproject,“ Duben 2015. [Online]. Available: https://www.yoctoproject.org/docs/1.8/ref-manual/ref-manual.html#ref-classes-useradd. [Přístup získán 25 Duben 2019].