Univerzita Komenského, Bratislava
Fakulta Matematiky, Fyziky a Informatiky
Katedra Informatiky

Evidenčné číslo: bbe8f1a2-aea9-433a-8289-d437be661f6f

Virtuálne zariadenia

Generátor Linux-ových distribúcií

(Diplomová práca)

Študijný program: Informatika
Študijný odbor: 9.2.1 Informatika
Školiteľ: RNDr. Richard Ostertág, PhD.

Bratislava, 2011 Peter Bolemant

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Peter Bolemant
Študijný program: informatika (Jednoodborové štúdium, magisterský I.II. st., denná forma)
Študijný odbor:
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Generátor linuxových distribúcií pre virtuálne zariadenia

Cieľ: 1. Vytvoriť generátor pre linux distribúcie pomocou user-friendly rozhrania (napr. cez web) so zameraním na minimálnu veľkosť a optimálnosť výslednej distribúcie s prihliadnutím na jej využitie ako guest operačného systému pri virtualizácii. 2. Generátor distribúcie by mal minimálne podporovať voľbu kernel-u, bootloader-a, shell-u, package manager-a, balíčkov. 3. Jednotlivé časti by mali byť konfigurovateľné pomocou konfiguračných šablón. 4. Výstupom generátora bude image funkčnej distribúcie. 5. Porovnanie výkonu, veľkosti a iných parametrov vytvorených distribúcii pri rôznych konfiguráciách a aj s inými existujúcimi distribúciami pre virtuálne stroje.

Vedúci: RNDr. Richard Ostertág, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 16.11.2006

Dátum schválenia: 04.05.2011

Mgr. Tomáš Plachetka, Dr.
garant študijného programu

.....
študent

.....
vedúci

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne iba s použitím citovaných zdrojov.

. . . . . .

Ďakujem školiteľovi mojej diplomovej práce RNDr. Richardovi Ostertágovi, PhD. za konzultácie a cenné rady počas jej tvorby. Tiež ďakujem RNDr. Jaroslavi Janáčkovi, PhD. za technickú pomoc a v neposlednom rade kolegom a priateľom za trpezlivosť a psychickú pomoc počas písania práce.

Abstrakt

Autor: Peter Bolemant
Názov práce: Virtuálne zariadenia: Generátor Linux-ových distribúcií
Škola: Univerzita Komenského v Bratislave
Fakulta: Fakulta matematiky, fyziky a informatiky
Katedra: Katedra informatiky
Školiteľ: RNDr. Richard Ostertág, PhD.
Rok: 2011
Rozsah práce: 80 strán

Témou a cieľom tejto diplomovej práce je vytvorenie online generátora Linux-ových distribúcií s automatickým inštalátorom minimalizujúcim potrebu zásahu používateľa po jej vytvorení. Časť práce sa venuje prehľadu Linux-ových systémov ako takých. Rozoberieme súborové systémy podporované Linux-om a jednotlivé komponenty Linux-ových systémov ako je kernel, adresárová štruktúra, shell, booter a nakoniec aplikácie; vytvoríme si tak celkový prehľad o tom, čo má byť výsledkom skladania Linux-ovej distribúcie. V ďalších častiach práce vysvetlíme postup manuálnej tvorby distribúcie a uvedieme známe online aplikácie umožňujúce jej generovanie. Prácu uzatvára navrhovaný spôsob riešenia uvedeného cieľa vo forme novej online aplikácie generatux.

Kľúčové slová: Linux, distribúcia, online generovanie, automatizácia, generatux

Abstract

Author: Peter Bolemant
Title: Virtuálne zariadenia: Generátor Linux-ových distribúcií
University: Comenius University in Bratislava
Faculty: Faculty of Mathematics, Physics and Informatics
Department: Department of Computer Science
Supervisor: RNDr. Richard Ostertág, PhD.
Year: 2011
Number of pages: 80

The theme and goal of this diploma thesis is to create an online generator of Linux distributions aimed towards their installation automation in order to minimalise the need of user's intervention. In the first part, that is dealing with Linux systems in general, we will take a look on filesystems supported by Linux as well as on the components Linux contains, namely the kernel, FHS – filesystem hierarchy standard, the shell, the booter and application layer. That will lead us to the next section explaining the manual distribution building process. Then we will try to compare existing online applications dealing with similar tasks in order to proceed to the last section of this thesis – the generatux application, which is supposed to solve the given problem.

Keywords: Linux, distro, online generation, automatisation, generatux

Obsah

Kapitola 1

Úvod

Pod Linux-ovou distribúciou chápeme vopred pripravenú a predkonfigurovanú množinu aplikácií, pričom celý systém je založený na Linux-ovom jadre. Linux, voľakedy chápaný ako operačný systém určený len profesionálom pôsobiacich vo sfére informačných technológií, ale nevhodný pre bežné, kancelárske použitie, sa v súčasnosti začína presadzovať aj v tejto oblasti. Je to najmä vďaka vylepšenému grafickému používateľskému rozhraniu a širokému spektru aplikácií. Tieto boli spočiatku tvorené nadšencami, študentami; jedinou reguláciou bola ich vzájomná komunikácia. Ako vývoj postupoval, narastala potreba pre štandardizáciu práce a centralizovanie sa do vývojárskych tímov. Štandardy, ktoré vznikli však (našťastie) neboli dostatočné striktné na to, aby týmto tímom neumožňovali vydávať Linux-ové distribúcie „šité“ na mieru za nejakým konkrétnym účelom, či pre nejaké konkrétne zariadenie. Tieto distribúcie boli spočiatku tvorené manuálne a predstavovali zložitý proces, ktorý bol pre bežného používateľa nezrealizovateľný.

Jedným z dôvodov prečo Linux nebol natoľko rozšírený boli tiež náklady s tým spojené. Ak bol používateľ obmedzený na jeden počítač a zvyknutý na na ňom nainštalovaný operačný systém, pre vyskúšanie ďalšieho systému, ak nechcel byť obmedzovaný jeho prevádzkou, bol jednoducho potrebný dodatočný hardvér. Momentálne sú ale populárne aplikácie, ktoré tento problém

odstraňujú virtualizáciou – simulujú hardvér na danom hostiteľskom systéme, ktorý umožní paralelný chod ďalšieho operačného systému.

Cieľom tejto práce je ale ukázať, že v súčasnosti je možné Linux-ovú distribúciu vytvoriť online, pomocou webového rozhrania, s postačujúcou minimálnou množinou znalostí o jej reálnom skladaní. Navyše ukážeme, že týmto spôsobom je možné vytvoriť plne automatizovanú inštalačnú aplikáciu, ktorá vytvorí kompletný systém s naformátovaným diskom a nainštalovanou distribúciou. Výsledkom teda bude online systém, ktorý umožní jednoduchým, intuitívnym spôsobom konfigurovať atribúty takejto aplikácie a vytvorí bootovateľný ISO obraz. Pre jeho samotný vývoj je však potrebné vykonať analýzu Linux-u, distribúcií a ich tvorby. Motiváciou pre takýto cieľ je zjednodušiť používateľom celý proces inštalácie na samotnom cieľovom hardvéri, kedy bude postačovať iba stlačenie jednej klávesy. Virtualizovaný systém zjednoduší celý proces vývoja, pretože zabezpečí vhodné testovacie prostredie.

Zvyšok tejto kapitoly sa zaoberá historickými momentami vzniku Linux-u ako takého a poskytuje čitateľovi podrobnejší úvod do problematiky.

V druhej kapitole predstavíme štruktúru Linux-u. V jej prvej časti popíšeme funkcionalitu jadra – kernelu, zhrnieme jeho výhody, nevýhody a dôležité funkcie. Ďalej porovnáme súborové systémy, ktoré Linux podporuje so zameraním sa na bežne používané, ale uvedieme aj súborové systémy, ktoré sú zatiaľ vo vývojovom štádiu, ale predpokladá sa ich úspech. Následne vysvetlíme čo to je virtuálny súborový systém, ako Linux pracuje s adresárovou štruktúrou a ako táto štruktúra vyzerá. Popíšeme význam jednotlivých dôležitých adresárov a vlastnosti súborov v Linux-e. V ďalších častiach tejto kapitoly si predstavíme základné rozhranie pre prácu s Linux-om – shell, porovnáme niektoré jeho používané aplikácie a zhrnieme proces nabootovania a štartu Linux-ového systému. Záver kapitoly sa bude venovať tomu, čo tvorí charakter distribúcie – aplikáciám a spôsobe akým s nimi Linux manipuluje.

Tretia kapitola sa bude zaoberať manuálnym procesom tvorby Linuxových distribúcií. Popíšeme jeho jednotlivé kroky a predstavíme základné aplikácie potrebné pre tento proces. Posledná časť tejto kapitoly sa venuje inštalačnému procesu distribúcií vo všeobecnosti.

Predposledná kapitola vytvorí úvod do možností generovania distribúcii pomocou online systémov. Budú predstavené vybrané tri známe existujúce systémy, ich metódy riešenia danej úlohy a funkcionalita, ktorú poskytujú.

A nakoniec, v záverečnej kapitole sa budeme venovať samotnému riešeniu daného cieľa; popíšeme ako vytvorený systém funguje, aké distribúcie generuje a čo presne je jeho výstupom.

1.1 História UNIX-u a UNIX-ových systémov

1.1.1 Stručná história Unix-u

Počiatky Unix-u siahajú do polovice 60-tych rokov 20-teho storočia, kedy vznikla myšlienka o systéme s nepretržitou prevádzkou, dostupnom kedykoľvek, s vysokým stupňom škálovateľnosti a podporou viacerých používateľov. Tento projekt, nazvaný Multics (Multiplexed Information and Computing Service), bol spoločnou snahou popredných amerických technologicky zameraných spoločností a univerzít, najmä AT&T Bell Labs a MIT, a z veľkej časti financovaný americkým ministerstvom obrany, resp. dcérskou agentúrou DARPA (Defense Advanced Research Projects Agency), čoho dôsledkom bola ďalšia požiadavka a to bezpečnosť alebo inak – ochrana údajov. [SG03]

Príčinou neúspechu sa však paradoxne stalo práve to, čo si tvorcovia od Multics-u sľubovali. Požiadavky boli na tú dobu jednoducho príliš silné a vývoj vôbec nenapredoval želaným tempom a tak v roku 1969, AT&T Bell Labs projekt Multics zrušili. No našťastie, jeho vývojári, hlavne Ken Thompson, Dennis Ritchie, Peter Neumann (ktorý navrhol názov nového operačného systému – Unix) a Doug McIlroy, sa rozhodli pokračovať samostatne, bez

podpory, avšak v inom duchu. Páčila sa im interaktivita a prostredie systému Multics, no snažili sa o celkové zjednodušenie s cieľom dosiahnuť menšie hardvérové nároky a hlavne neupriamovať toľkú pozornosť na bezpečnosť, ale na funkcionalitu, teda schopnosť spúšťať programy. Týmto počinom však zároveň vzniklo čosi viac ako len niečo technické. D. Ritchie to opísal slovami: „Čo sme chceli zachovať, nebolo iba kvalitné programátorské prostredie, ale systém, okolo ktorého sa môže utvoriť spoločenstvo. Zo skúseností vieme, že základ tímového vývoja cez vzdialený prístup nie je iba o písaní programov do terminálu miesto dierných štítkov, ale o úzkej komunikácií.“ [Ray03, Neg06]

Zrodila sa tak spolupráca, ktorej výsledkom bol neskôr vznik Unix-u ako operačného systému. Jeho vývoj autori spočiatku maskovali vývojom aplikácie pre spracovanie textu, pretože vedenie AT&T Bell Labs nemalo v pláne vývoj nového vlastného operačného systému, no keď ho Thompsonov tím predstavil hotový, vzbudil záujem a s vidinou úspechu ho začali AT&T Bell Labs finančne podporovať. Idea využitia spočívala v balíku menších jednoduchých programov – nástrojov, pričom ak chcel používateľ komplikovanejší výsledok alebo spracovanie vstupu, získal ho využitím alebo skôr spojením týchto nástrojov. Ďalši pokrok nastal v roku 1973 vo forme prepisu kódu do programovacieho jazyka C (predtým assembler pre PDP 11/20) napriek tomu, že bola zastávaná myšlienka, že tak komplexný systém ako je operačný, musí byť napísaný v assembleri. Obhajobou prepisu bol fakt, že Unix sa tak stal takmer nezávislým na cieľovom hardvéri. [SG03, Ray03, Neg06]

Prvé informácie o Unix-e, Thompson a Ritchie zverejnili koncom roku 1973. Unix verejnosť zaujal, avšak kvôli rozhodnutiu protimonopolného úradu, AT&T Bell Labs malo zákaz vstupovať na počítačový trh, nebolo možné Unix distribuovať ako produkt, no jeho tvorca sa rozhodol vyhovieť požiadavkám záujemcov (predovšetkým univerzity a výskumné ústavy) a diskrétne Unix odosielal, podľa legendy, s venovaním „s láskou, ken“. V rozmedzí niekoľkých mesiacov sa Unix rýchlo šíril a v roku 1977 bolo odhadovaných viac ako

500 inštalácií. Nadšenie z Unix-u vrcholilo v množstve rozšírení pôvodného systému o menšie programy a vylepšenia, ako napríklad podpora viacerých používateľov, vznikajúcich prednostne na univerzitnej pôde. Väčšina týchto rozšírení bola implementovaná v oficiálnom siemdom vydaní Unix-u od Bell Labs v roku 1979, no hlavné centrum diania zostávalo naďalalej v univerzitách, primárne na Kalifornskej Univerzite v Berkeley. Na oblasť bezpečnosti systému to však malo negatívny vplyv, pretože akademické laboratóriá mali obmedzený prístup samé o sebe a tak nebol dôvod rozširovať Unix týmto smerom. [Ray03, Neg06]

Ďalším veľkým krokom vo vývoji Unix-u bola seriózna implementácia sieťovej komunikácie. Podiel na ňom má práve univerzita v Berkeley, keď na žiadosť americkej agentúry DARPA doplnila do ich verzie – BSD (Berkeley Software Development) Unix-u implementáciu množiny sieťových protokolov TCP/IP a tak vznikla v roku 1983 verzia BSD Unix 4.2, ktorá spojila svet Unix-u a predchodcu internetu – ARPANET-u. Dovtedy mal Unix len veľmi slabú podporu sieťovej komunikácie. Šlo len o jednoduchú distribúciu dát cez telefónne linky (neskôr program Usenet – sofistikovanejšie, avšak stále nie postačujúce). Univerzita v Berkeley sa však zaslúžila aj o ďalšie úspechy – možnosť používateľa prepínať medzi súčasne bežiacimi aplikáciami a tiež podpora dlhých názvov súborov (až 255 znakov oproti 14 znakom v Unix-e od AT&T Bell Labs). [Ray03, Neg06]

V tejto dobe tiež začala vyvíjať aktivitu aj firma Sun, ktorej spoluzakladateľom bol Bill Joy, vedúci tímu, ktorý je zodpovedný za prvé vydanie BSD. Spojením hardvéru navrhnutom na Standfordskej Univerzite a Berkeley Unix-u sa vývojarom Sun-u podaril veľký úspech na poli pracovných staníc a začali komercionalizovať svoje výsledky. V roku 1983 však nastala dôležitá udalosť. Protimonopolný úrad vyhral ďalší spor s AT&T Bell Labs, čoho dôsledok bolo oddelenie Bell Labs od AT&T a to im umožnilo vstúpiť s Unix-om ako komerčným produktom (Unix System V) na trh. Najprv sa zdalo, že reálna

konkurenčná rivalita bude mať dobrý vplyv na ďalší vývoj, no AT&T prestalo uvoľnovať zdrojový kód a akademické rozšírenia upadali na intenzite. Unix začal svoj úpadok. Sun, AT&T a iné spoločnosti, ktoré si začali Unix osvojovať a rozširovať, sa totiž dopustili niekoľkých veľkých strategických chýb. V prvom rade to bola snaha o získanie výhody vďaka rozdielom v jednotlivých distribúciách, z čoho vzniklo len viac druhov rozhraní a narušila sa podpora viacerých platform. Ďalšou chybou bolo podcenenie druhej vetvy výpočtového priemyslu a to osobné počítače, ktoré začal ovládať Microsoft svojim operačným systémom MS-DOS a to i napriek tomu, že Intel vydal svoj procesor s označením 80386, ktorý najmä vďaka väčšiemu adresnému priestoru už bol schopný chodu Unix-u. No aj Microsoft, aj keď to pre nich nebolo prioritné, sa začal Unix-om zaoberať a vydával ho pod názvom Xenix. [Ray03, SG03]

Takto začalo obdobie ľudovo nazvané vojna Unix-ov (Unix Wars). Rozpor sa dá vnímať aj z hľadiska technickej orientácie ľudí, ktorí sa na ňom podieľali. Programátori a technici viac presadzovali BSD Unix oproti obchodníkom, ktorí sa upriamovali na System V. No nastali aj progresívne zmeny. Bol vyvinutý jednoduchý avšak veľmi nápomocný program `patch(1)`, ktorý umožnil inkrementálne aktualizácie systému. Takto sa vývoj mohol rozdeliť do viacerých vetiev – oblastí, vykonávať paralelne a znova spojiť. V 90-tych rokoch to spôsobilo revitalizciu Unix-u vďaka kolaboratívnej práci s využitím internetu. Avšak počas tohto obdobia, konkrétne v roku 1985, vznikol revolučný dokument alebo skôr filozofia vývoja a distribúcie softvéru. Šlo o projekt Richarda Stallmana nazvaný GNU (GNU's Not Unix), ktorého snahou bolo vybudovať slobodný operačný systém – otvorený zdrojový kód a voľná distribúcia, so všetkými (aj pre vývojára, aj pre bežného používateľa) potrebnými aplikáciami, kvôli čomu vznikla aj zastrešujúca organizácia Free Software Foundation. Avšak aj keď sa vývojárom GNU podarili úspechy v podobe C kompilátora a koncom roku 1987 mali hotovú základnú sadu používateľských a

vývojových nástrojov (najmä kompilátor, debugger, editor), dopustili sa rovnakej chyby ako vývojári komerčného Unix-u a to neupriamenie pozornosti na platformu PC. Taktiež v tomto období, kompiláciou spoločných čŕt systémov BSD a System V Release 3, vznikol dôležitý štandard POSIX (pôvode inicializovaný IEEE pod označením Std 1003.1-1988) dávajúci základ všetkým nasledujúcim systémom postavených na Unix-ovom jadre. POSIX (Portable Operating System Interface) je špecifikáciou rozhrania medzi používateľom či aplikáciami a operačným systémom; definuje systémové volania, knižničné funkcie a chovanie programov. [Ray03, SG03]

Zvyšok 80-tych rokov minulého storočia však bol pre Unix nepríjemný. Microsoft slávil úspech na poli osobných počítačov a trhu menších spoločností, no Unix upadal. Hlavnou príčinou bola aj orientácia na pracovné stanice, no primárne neuvoľnenie kódu, striktná komercionalizácia a aj napriek POSIX štandardom, veľká heterogenita v distribúciách evokujúca chaos. V týchto rokoch dokonca vznikla akási opozícia, nesúhlas s Unix-om, v podobe Open Software Foundation založenej výrobcami hardvéru – IBM, DEC, Hewlett-Packard a inými, pričom jej vznik bol motivovaný najmä strachom zo spojenia AT&T a Sun. Tieto dve spoločnosti sa totiž kvôli uvedeným problémom nevedeli samostatne s vývojom pohnúť ďalej. Počiatočné nadšenie v Unix-ovej komunite vystriedal zármutok a sklamanie, spoluprácu vystriedala rivalita a zmätok a tak dobehnúť rapídne narastajúci Microsoft bolo nemožné. Ďalším negatívnym momentom bol neúspech projektu GNU vydať vlastné Unix-ové jadro, čo by mohlo Unix-ovú komunitu opäť postaviť na nohy, no keďže sa nič podobné nedialo, vierohodnosť tohto projektu upadala. [Ray03, SG03]

Situácia na univerzitnej pôde bola predsalen odlišná. Keďže došlo k veľkým reštrikciám a kontrole autorských práv, vznikla snaha vyčistiť BSD od pôvodného kódu AT&T, čo znamenalo preprogramovanie jednotlivých častí len na základe ich špecifikácie. Po niekoľkých iteráciách zostala len malá časť chráneného kódu, ktorú ale William Jolitz, jeden z vývojárov BSD, prepísal pre

8086 architektúru a vznikol tak nový systém (prozaicky nazvaný 386/BSD). To sa dialo na konci roku 1991. Niekoľko mesiacov neskôr tento úspech zjednotil pár ľudí do projektu známeho pod menom NetBSD, no už po krátkej chvíli došlo k nezhode akým smerom sa uberať. Jedna skupina mala tendneciu tento systém rozšíriť na čo najviac platforiem, kým druhá sa upriamovala na základnú 8086 platformu a šlo im o vylepšovanie systému ako takého. Druhá skupina sa preto oddelila a svoj projekt pomenovala FreeBSD, no názory sa neskôr opäť rozdelili a vznikol ďalší projekt resp. frakcia – OpenBSD, ktorý mal byť spoľahlivejší, tj. stabilnejší a bezpečnejší. [Ray03, SG03, Neg06]

1.1.2 Stručná história Linux-u

Linus Torvalds, vtedy ešte študent, inšpirovaný systémom MINIX, čo bol akademický pokus o otvorený Unix-ový operačný systém, začal projekt – Linux. Páčil sa mu MINIX ako systém, ktorý používal v škole a chcel mať niečo podobné aj na svojom počítači, avšak komplexnejšie, pretože MINIX bol príliš jednoduchý a slabo výkonný. MINIX vytvoril profesor na amsterdamskej univerzite Vrije, Andrew S. Tannenbaum, pričom medzi jeho najväčšie pozitíva patrí veľmi dobrá dokumentácia; bol voľne prístupný a schopný behu na 8086 architektúre, ale kvôli spomenutým nevýhodám nebol pre masové nasadenie, a už vôbec nie pre komerčnú sféru, použiteľný. Jednoducho bol vytvorený a vhodný len pre výuku. Keďže počítače s 8086 architektúrou alebo inak – PC kompatibilné boli v tej dobe už cenovo dostupné bežným domácnostiam a Linus nemal možnosti vyvíjať pre inú architektúru, táto platforma sa stala počiatočnou pre návrh Linux-u. [SG03, Neg06, Tho05]

Linus začal tvoriť svoj operačný systém ako koníček, pre svoje potešenie. Prvé reálne výsledky mal už v auguste 1991, v roku kedy s projektom začal, vo forme prototypu, respektíve základnej verzie 0.01. Najdôležitejším počinom, ktorý viedol k dnešnej popularite Linux-u bolo, podľa môjho názoru, využitie v tej dobe sa rozširujúceho internetu, ktorý umožnil Linusovi

oslovovať ďalších vývojárov, ktorí mu so systémom pomáhali nielen nápadmi, ale aj vlastným kódom, hlavne ovládačmi pre hardvér a testovaním; a to všetko na dobrovoľníckej báze bez nároku na honorár. Ďalším dôležitým aspektom bolo licencovanie Linux-u. Od verzie 0.99 bol vydaný pod GNU GPL (General Public License) licenciou. To znamená, že ktokoľvek mal prístup ku kódu, mohol ho upravovať a rozširovať, no licencia musela zostať zachovaná. Okolo Linux-u sa tak utvorila komunita nadšencov, podobne ako to bolo v prípade počiatku Unix-u, no rozdiel bol v tom, že doba umožňovala oveľa lepšiu vzájomnú komunikáciu a koordináciu a to vývoju napomáhalo ako v rýchlosti, tak aj v kvalite. Už spomenutý program `patch(1)` sa pri vývoji osvedčil ako veľmi spoľahlivý nástroj. Veľkým zlomom v prospech úspechu bolo tiež vydanie dodnes úspešného web servera Apache, ktorý dokázal bez problémov konkurovať proprietárnym riešeniam, keďže bol rovnako ako Linux samotný – stabilný, výkonný a navyše voľne licencovaný. Postačoval aj pre náročné projekty a tak sa v kombinácií s Linuxom rýchlo stali základom pre väčšinu webových aplikácií. [Tho05, Ray03, Chi04]

To, čo Linus vytvoril, bol však „iba“ kernel operačného systému, no pre kompletný systém potreboval taktiež shell, kompilátor, knižnice a aplikácie. Rozhodol sa teda využiť tieto prostriedky z GNU (ako kompilátor použil dodnes hádam neprekonaný Stallmanov GNU Compiler Collection – gcc) a tak sa dalo tušiť, že spolupráca medzi Linuxom a GNU bude užšia. Vývojári projektu GNU stále nemali vlastný operačný systém, aby mohol byť GNU kompletný, avšak na druhej strane im Linux vyhovoval a tak, vzhľadom na otvorenosť kódu a možnosť priamej spolupráce s Linusom, začali pre Linux písať ďalšie programy, ktoré boli priamo integrované v inštalačných balíkoch, nazývaných distribúciami. [Tho05, Ray03, Chi04]

Ako vývoj pokračoval, Linux bol portovaný aj na iné platformy ako na pôvodnú PC architektúru a začal byť dokonca používaný aj na veľmi výkonných serveroch, resp. superpočítačoch. Na druhej strane taktiež na výkon-

nostne veľmi obmedzených zariadeniach ako napríklad mobilných PDA či v inteligentných telefónoch. Ukázalo sa tak, že je nielen stabilný, ale aj veľmi dobre prispôsobivý. Postupom času Linus odstupoval od samotného programovania a venoval sa skôr spravovaniu vývoja jadra, vyvíjaného ostatnými programátormi. Ďalšia dôležitá verzia kernelu bola vydaná Alanom Coxom v roku 1999 vo verzii 2.2. Bola to prvá stabilná verzia, ktorá umožňovala priamy výber výrobcu a typu procesora a tak zabezpečovala lepší, optimalizovaný výkon, vzhľadom na cieľovú achitektúru. Tiež obsahovala lepšiu podporu pre vstupno-výstupné zariadenia a rôzne systémy súborov, ako aj NTFS od Microsoft-u. Väčší prínos mala verzia 2.4 (Marcelo Tosatti, 2001) s podporou dôležitých štandardov, respektíve rozhraní USB a PCMCIA; neskôr, v ďalších podverziách prišla podpora Bluetooth a RAID. Ďalšia dôležitá verzia 2.6 bola vydaná koncom roku 2003 a je doteraz udržiavaná samotným Linusom Torvaldsom. Poslednou, aktuálnou stabilnou verziou kernelu je 2.6.38.4 (27.4.2011). [Tho05, Wikl]

Kapitola 2

Linux a jeho architektúra

To, že je Linux natoľko populárny a rozšírený aj v kritických oblastiach, kde malá chyba môže znamenať nielen finančnú stratu, ale aj zdravotnú ujmu svedčí o jeho kvalite. Na vývoji sa podieľajú jedni z najlepších programátorov. Linux ako taký je naprogramovaný v jazyku C a skompilovaný kompilátorom GCC, pretože to bol jediný kompilátor vyhovujúci Linusovmu kódu. Ďalšími atribútmi úspechu Linuxu sú široká portabilita na rôzne platformy, stabilita, veľký výkon aj pri veľmi malom jadre, ktoré sa aj s niekoľkými jednoduchými aplikáciami zmestí na 1,44" disketu. Nezanedbateľným faktom je aj to, že Linux nie je komerčný a teda je bez akýchkoľvek poplatkov (toto však už nezahŕňa jednotlivé distribúcie a podporu).

Dôležitým aspektom, kvôli ktorému si Linux vybudoval spomenuté atribúty úspechu je jeho architektúra. Linux alebo výstižnejšie GNU/Linux, keďže Linux je priame pomenovanie pre jadro celého operačného systému, by sa dal rozdeliť do nasledujúcich úrovní:

- Jadro – kernel
- Systém súborov
- shell

- Aplikácie
- Používatelia

Pozrieme sa teda bližšie na každú z týchto častí.

2.1 Kernel

Kernel – jadro, je hlavná časť všetkých operačných systémov. Je to rozhranie medzi aplikáciami a hardvérom a teda slúži ako abstraktná vrstva pre programy. Jeho úlohou je správa prostriedkov, najmä prideľovanie prostriedkov procesom (procesy v Linux-e bývajú tiež často nazývané úlohami) a riadenie komunikácie medzi procesmi (IPC – interprocess communication). Kernel je teda nielen kľúčovým prvkom pre beh celého operačného systému, ale aj aplikácii nad ním a jeho návrh a implementícia sú kritickými faktormi výkonu, spoľahlivosti a bezpečnosti. Medzi prostriedky, ktoré kernel spravuje radíme procesor, pamäť a vstupno-výstupné zariadenia. Ak je kernel zavedený do pamäte a spustený, stáva sa taktiež procesom.

Ako už bolo spomenuté v predošlej kapitole, návrh a vývoj kernelu bol z veľkej miery inšpirovaný Unix-om, no má aj svoje odlišnosti. Kernel Linux-u totiž nebol navrhnutý podľa nejakého špecifického Unix kernelu, ale zámerom bolo zlúčenie dobrých vlastností rôznych Unix-ových kernelov. Súčasné jadro vo verzii 2.6 však už spĺňa väčšinu z množiny štandardov IEEE POSIX (Portable Operating Systems based on Unix), čo dáva predpoklad, že programy písané pre Unix budú vo veľkej miere skompilovateľné a spustiteľné aj na Linux-e. Množina štandardov POSIX totiž definuje API (Application Programming Interface), teda rozhranie, cez ktoré by mali programy, pokiaľ majú byť kompatibilné s Unix-ovým kernelom, komunikovať so systémom. [Lov05]

Z technického hľadiska je kernel Linux-u monolitický. To znamená, že

celý systém je spustený v kernel móde a používateľský mód je určený len pre aplikácie. Kernel mód je úzko spätý s hardvérom – procesorom a periférnymi zariadeniami. Procesy bežiace v tomto móde majú priamy prístup k systémovým rutinám a dátovým štruktúram, čo sú napríklad aj ovládače zariadení, naproti procesom bežiacim v používateľskom móde, ktoré k týmto prístup štandardne nemajú. Ak chce proces bežiaci v používateľskom móde použiť zariadenie alebo volať nejakú systémovú rutinu, kernel mu to najprv musí umožniť tým, že procesu dočasne povolí prechod do kernel módu. [DPB05, Wikl]

Je sporné, či bola monolitická architektúra vhodná voľba, keď už v tom čase boli obhajované novšie trendy, napríklad mikrokernel. Linus viedol na túto tému debatu priamo s profesorom Tannenbaumom, ktorý považoval Linux za zastaralý (Tannenbaum debatu začal príspevkom s názvom: „Linux is obsolete“ [Tan]). Upozorňoval hlavne na slabé možnosti portability monolitického kernelu. Linux bol totiž cielený pre x86 architektúru, ktorej Tannenbaum nedával budúcnosť vzhľadom na vysokú cenu. No v skutočnosti táto architektúra získala väčšinový podieľ na trhu a Linux sa podarilo portovať na veľa platforiem. Profesor Tannenbaum musel pripustiť, že sa v tejto veci mýlil. Napriek tejto „nevýhode“ je monolický kernel jednoduchší z hľadiska vývoja a vo všeobecnosti zabezpečuje lepší výkon, keď je dôležitý výkonný kód koncentrovaný do jedného celku. [Wikn]

Dôležitou vlastnosťou kernelu Linux-u je však možnosť dynamického nahrávania a uvoľňovania kernel modulov (LKM – Loadable Kernel Module) za behu systému na základe aktuálnej potreby, bez nutnosti ho reštartovať, ako je bežné napríklad pri systémoch Windows. Kernel moduly sú akési rozšírenia základného kernelu, ktoré v princípe nie je potrebné mať v pamäti neustále, respektíve je to nevýhodné. [Proc]

Slúžia ako:

- rozšírenia sieťových volaní

- interpretery programov
- ovládače zariadení
- ovládače súborových systémov
- sieťové ovládače

Výhod takéhoto modulárneho systému je hneď niekoľko. Jednou z nich sú nízke pamäťové nároky, keďže v pamäti je potrebný len základný kernel. Ďalšou výhodou je menšia náchylnosť na chyby pri kompilovaní kernelu, pretože jeho kód štandardne obsahuje len to najnutnejšie a praxou overené – menej kódu menšia pravdepodobnosť chyby. S tým súvisí aj diagnostika chýb – ak nastane nahratie nejakého modulu a systém následne prestane byť funkčným, je veľmi pravdepodobné, že chyba sa vyskytuje v tomto module. Ďalšou dôležitou vlastnosťou je možnosť konfigurácie kernelu pri jeho kompilácii.

Nevýhodou môže byť len potenciálna fragmentácia pamäte pri nahrávaní a odstraňovaní modulov, ale pri súčasných pamäťových kapacitách je to irelevantné.

Hlavné črty Linux kernelu (verzia 2.6) sú preemptívny multitasking, multithreading a podpora viacerých jadier.

2.1.1 preemptívny multitasking

Preemptívny multitasking rieši prepínanie medzi procesmi formou plánovača procesov a prerušeniami. Princíp tejto metódy spočíva v prideľovaní času na procesore kernelom a nie je riadený samotnými procesmi, aj keď kernel je v zásade tiež proces, no inak klasifikovaný. Procesy sa sa v tomto prípade neuvoľnujú samé, ale sú od procesora odoberané, čím sa predchádza možným konfliktom. [Lov05, DPB05]

2.1.2 multithreading

Kernel Linux-u podporuje aplikácie používajúce viac vlákien. Vlákna (angl. threads) sú akési takmer nezávislé podprocesy bežiaceho procesu. Môžu mať rôznu funkcionalitu a zodpovednosť, no zároveň môžu používať rovnaký adresný priestor, dátové štruktúry (globálne a statické premenné), zdielať prostriedky, zariadenia, a naviac majú rovnaký identifikátor procesu. [Lov05, DPB05]

Dôvodov, pre tvorbu nového vlákna oproti tvorbe nového procesu je viacero:

- tvorba vlákna je rýchla, rýchlejšia ako tvorba nového procesu
- prepnutie kontextu medzi vláknami je jednoduchšia operácia a trvá kratší čas
- zdieľanie údajov medzi vláknami je priamočiare

2.1.3 podpora viacerých jadier

Linux zabezpečuje takzvaný symetrický multiprocessing (SMP). Ide o prerozdeľovanie času na jednotlivých procesoroch alebo jadrách procesorov s čo najväčšou možnou mierou „spravodlivosti“. Teda každý proces alebo vlákno môže byť vykonávané na ľubovoľnom procesore. Kernel udržiava pre každý každé jadro procesoru dve fronty (angl. runqueues) s procesmi, respektíve vláknami – aktívnu a expirovanú. Ak procesu, či vláknu je odobratý čas na procesore, tento prechádza z aktívnej fornty do expirovanej. Takto má kernel prehľad o práci jednotlivých pocesorov, či jadier a pomocou vyrovnávača záťaže ju môže v cykloch rovnomerne distribuovať. [Lov05, DPB05]

2.2 Systém súborov

Pre uchovávanie dát, či už používateľských alebo aplikačných, je potrebné mať nejaké úložisko a toto úložisko nejakým spôsobom organizovať, manipulovať s ním a ochraňovať. Pre účely uchovávania dát slúžia fyzické pamäťové zariadenia, no zvyšok je pod kontrolou operačného systému. Ako pamäťové zariadenie – médium budeme v ďalšom uvažovať pevný disk. Dáta sú spravidla uložené vo forme súborov s rôznou povahou. To, akým spôsobom sú súbory organizované napríklad z hľadiska hierarchie, reštrikcií, či konvencií ohľadom mien súborov a adresárov (povolené znaky, koncovky súborov), prístupu, nazývame systémom súborov.

Aby bol pevný disk použiteľný, je potrebné ho najprv rozdeliť na jednu alebo viac partícií, čo sú logické oblasti disku použiteľné pre nejaký systém súborov. Partícia je pre daný typ systému súborov homogénna a teda nie je možné mať na partícií viac ako jeden systém súborov. Na druhej strane, je možné pevný disk rozdeliť na viacero partícií s rozdielnym systémom súborov na každej z nich. [Gar08]

Rozdeliť disk na viac partícií vo všeobecnosti prináša výhody z nasledovných dôvodov:

- **viacero operačných systémov** – viacero partícií umožnuje mať na jednom fyzickom počítači viacero rôznych operačných systémov – toľko, koľko je partícií.

- **bezpečnosť a aktívna prevencia** – v prípade viacerých partícií je tieto možné jednotlivo nastaviť a priradiť im význam (napríklad vyhradiť jednu partíciu pre súbory len na čítanie (read-only), súborom na ďalšej partícií jednotne zakázať ich spustenie pre nejakú skupinu používateľov a podobne), čo podporí zabezpečenie proti zneužitiu alebo odcudzeniu údajov. Môžeme takto tiež oddeliť kernel a blízke aplikácie, či knižnice

operačného systému od používateľských aplikácií a údajov a diverzifikovať tak zálohovanie.

- **ochrana údajov** – môže nastať situácia, že súborový systém nejakej partície bude poškodený. V tom prípade však škody, s najväčšou pravdepodobnosťou, nebudú siahať za hranice tejto partície a ostatné partície a hlavne ich obsah – teda dáta, ktoré môžu mať obrovskú hodnotu, zostanú nepoškodené.

- **ochrana systému** – veľkosť súborov, napríklad kde sa ukladajú informácie o behu systému a aplikácií (logovacie súbory) a počet dočasných súborov (tie zväčša nebývajú veľké, ale dominujú počtom) môže narastať, až sa súborový systém zaplní. To za predpokladu, že sa nachádzajú na rovnakej partícií ako samotný kernel alebo iné dôležité funkčné časti operačného systému môže spôsobiť neočakávateľné, ale hlavne neželané následky.

- **swapovanie** – pri veľkých pamäťovch nárokoch súčasných aplikácií je potrebné mať sekundárnu, lacnú, aj keď pomalšiu pamäť, ktorú procesy využívajú pre dáta, ktoré sa do primárnej pamäte už nezmestia. Sú to väčšinou dáta, s ktorými procesy momentálne nepracujú, ale pravdepodobne budú žiadané v ďalších krokoch. Ako riešenie tohto problému je vyhradené miesto na disku – špeciálna swap partícia.

- **výkon** – rozdelenie veľkého diskového priestoru na partície má za následok zvýšenie výkonu, pretože príbuzné dáta sú takto na disku držané vo vzájomnej fyzickej blízkosti a teda sa skracuje čas presunu diskovej hlavy.

Existuje množstvo rôznych súborových systémov a veľa z nich aj Linux podporuje (rádovo desiatky – kompletný zoznam je dostupný na [Hin]). Najčastejšie sa však v Linux-ových systémoch vyskutujú:

- Ext2
- Ext3
- Ext4
- ReiserFS
- Reiser4
- XFS
- Btrfs

2.2.1 Porovnanie súborových systémov

Ext2

Súborový systém ext2 (second extended filesystem) bol v minulosti štandardným východiskovým súborovým systémom pre väčšinu Linux-ových ditribúcií. Je druhou verziou predošlého ext FS, ktorý bol vylepšením súborového systému používaným Minix-om, avšak nebol príliš vhodný pre svoje obmedzenia – maximálna veľkosť jedného súboru aj partície bola 2 GB, neexistovala podpora žurnálu a časových známok zmeny súborov, navyše často dochádzalo k fragmentácii disku. Preto došlo k jeho vylepšeniu v podobe systému ext2. Ten už podporuje partície do veľkosti 4 TB s maximálnou veľkosťou súboru 2 GB, no má aj ďalšie funkcie. Zabezpečuje napríklad rezervu pre root používateľa, aby mohol vykonať potrebné opatrenia, v prípade, že by došlo k zaplneniu systému inými používateľmi a ich aplikáciami. Tiež využíva kontrolné mechanizmy pre zachovanie integrity údajov (stav systému uložený v superbloku), no táto kontrola nemôže byť vykonaná počas jeho pripojenia (mount); a ochranu zmazaných dát pred neželaným opätovným prístupom (príslušné bloky sa vyplnia náhodnými dátami), ak to používateľ potrebuje. Keďže je tento systém starší, je pomerne dobre optimalizovaný a stabilný, no neponúka

ďalšie funkcie potrebné pre moderný svet, hlavne žurnál. Absencia žurnálu ale na druhej strane znamená menší počet operácií s diskom a predlžuje tak jeho životnosť. Stále sa tak používa napríklad pre USB pamäťové zariadenia alebo SSD disky. [DPB05, Neg06, oL]

Ext3

Systém ext2 bol navrhnutý tak, aby ho bolo možné ďalej rozširovať bez potreby formátovania a následnej straty nezálohovaných údajov. Ext3 (third extended filesystem) je rozšírením systému ext2 a teda konverzia na tento systém je z ext2 priamočiara (s menšími úpravami aj spätne – z ext3 na ext2). Táto črta je však sčasti aj nevýhodou, pretože ext3 sa nemôže od starého sytému funkčne a štrukturálne príliš vzďaľovať a tak neprináša veľa nového. Medzi najdôležitejšie vylepšenia patrí hlavne žurnál – metóda zabezpečujúca integritu dát pri neočakávaných udalostiach (výpadky, zlyhanie hardvéru) pomocou zapisovania informácii o zmenených údajoch pred ich samotným zápisom na disk a indexácia pomocou H-stromov. Naďalej si zachováva vyspelosť ako u jeho predchodcu a teda tento súborový systém je odporúčané používať za bežných okolností alebo keď sa používateľ nevie rozhodnúť aký súborový systém vybrať; voľbou ext3 nič nepokazí. Ext3 má aj zopár nevýhod: naďalej neumožňuje kontrolu integrity údajov počas pripojenia systému, tiež neumožňuje skutočnú defragmentáciu (systém sa na základe snahy o alokáciu najbližších blokov pri zápise spolieha na to, že ku výraznej fragmentácii nedôjde) a v neposlednom rade je to absencia kontrolných súčtov žurnálu pre dodatočnú kontrolu v prípade výpadkov. [DPB05, Sch04, Neg06]

Ext4

Ext4 (fourth extended filesystem) je ďalším pokračovaním rodiny súborových systémov ext. Vznikol na báze systému ext3 a je pomerne nový; považovaný za stabilný a podporovaný sa stal vo verzii kernelu 2.6.28 v roku 2008. Je

dopredne kompatibilný s jeho predchodcami, no spätné pripojenie systému ako ext2 alebo ext3 pri aktívnom použití hlavnej funkcionality systému ext4 už možné nie je. Ext4 zabezpečuje lepšiu alokáciu blokov a tým sa stáva efektívnejším. Maximálna podporovaná veľkosť partície pre ext4 je 1 EB so súbormi veľkosti najviac 16 TB. Ext4 oproti jeho predchodcom prináša tieto funkcie a možnosti:

- perzistentná predalokácia – pomocou systémových volaní; predalokáciou sa súborom zabezpečí včasné a trvalé dostatočné miesto na disku s predpokladom blízkych blokov.
- pozdržaná alokácia – systém sa snaží alokovať miesto až v čase, kedy bude súbor zapísaný na disk.
- hromadná alokácia – súvisí s predošlou funkciou. Ext3 alokoval bloky postupne, sekvenčne. Naproti tomu, v kooperácii s pozdržanou alokáciou, ext4 alokuje bloky v dávkach, naraz pre jeden súbor, čo má za následok menšiu fragmentáciu disku.
- kontrolné sučty žurnálu – slúžia pre zvýšenie spoľahlivosti žurnálu, keďže tento je tiež súbor a môže byť pri výpadku v nevhodnom čase poškodený.
- rýchlejšia kontrola integrity dát – systém ext4 udržiava zoznam nepoužitých blokov a tak program pre kontrolu partície, napríklad `fsck`, tieto nepoužité bloky pri kontrole môže vynechať, čím sa dosahujú väčšie rýchlosti kontroly.
- presnejšie časové známky – ako rastie rýchlosť procesorov a diskových zariadení, je nutnejšie presnejšie meranie času; ext4 preto meria čas v nanosekundách. To je výhodné napríklad pri ladení programov, keď chce používateľ zmapovať sekvenciu zápisu dát do viacerých súborov.

Najväčším prínosom však bolo zavedenie takzvaných extentov – zoskupení susedných blokov. Dovtedy boli bloky súboru mapované v zozname s adresami jednotlivých blokov, čo v mnohých prípadoch znamenalo zbytočné udržiavanie veľkého množstva adries. Následkom bola ďalšia kapacitná záťaž a pomalé spracovanie súborov. Pre prístup a manipuláciu s extentom však postačuje vedieť len jeho začiatok a počet blokov v ňom. Extenty majú tiež kapacitné obmedzenia – 128 MB a tak väčšie súbory sú rozdelené do viacerých extentov, ktoré sú v prípade, že ich počet presiahne 4, ďalej indexované H-stromom. Extenty sú tiež používané aj v iných moderných súborových systémoch; zvyšujú výkon systému a tým, že sú tvorené susednými blokmi, napomáhajú eliminovať fragmentáciu.

Aj keď je ext4 považovaný za stabilný, je stále pomerne novým súborovým systémom, avšak je odporúčaný pre použitie ako pre pracovné stanice, tak aj pre servre a teda pre nie striktne konzervatívneho používateľa je vhodnou voľbou. [Wiki]

ReiserFS

Tento súborový systém (tiež nazývaný Reiser3) je podobný ext3. Ako prvý mal zavedený žurnál a preto sa aj stal východiskovým súborovým systémom niekoľkých distribúcií (v minulosti aj populárny SUSE Linux, ktorý neskôr prešiel na ext3). Ďalšou jeho hlavnou črtou je dynamické rozširovanie priestoru podľa potreby. Pre indexáciu používa B+ stromy, ktoré sú vhodné pre množstvo malých súborov. Napriek týmto výhodám sa ReiserFS príliš neujal, pretože pri väčších súboroch jeho výkon nezanedbateľne klesá a tiež dosahuje zlé výsledky pri procesoroch s viac jadrami. [Sch04, Neg06]

Reiser4

Reiser4 má byť nasledovníkom ReiserFS písaným nanovo, riešiacim problémy, ktoré sa vyskytovali pri staršom systéme. Pre indexáciu používa B*

stromy, podporuje transakcie a všeobecne má vyšší výkon ako jeho predchodca; navyše podporuje zásuvné moduly, napríklad pre šifrovanie alebo kompresiu dát. Je však stále vo vývoji a tak je jeho budúcnosť vzhľadom na už používané a kernelom podporované súborové systémy neistá. V súčasnosti jeho použitie nie je odporúčané. [Wikm]

XFS

Ide o ďalší žurnálový systém; pôvodne vyvíjaný pre operačný systém IRIX spoločnosti Silicon Graphics International, no neskôr zabudovaný aj do kernelu Linux-u (od verzie 2.4 v roku 2001). Je 64-bitový, s maximálnou podporovanou veľkosťou partície 16 EB a súbormi s veľkosťou 8 EB, čo už sú čísla, ktorým ostatné súborové systémy príliš nekonkurujú. Navyše pri tvorbe partície je možné zvoliť veľkosť blokov, v závislosti od architektúry a kernelu, v rozmedzí od 512 B do 64 KB a tým zvýšiť výkon na základe toho, či je predpokladom veľa menších súborov alebo menej väčších a obsahuje prostriedky pre zväčšenie partície po jej vytvorení a používaní. Žurnál XFS je navrhnutý tak, aby bola zachovaná konzistencia a kontroly integrity boli veľmi rýchle, dokonca nezávislé od veľkosti systému. Zapisujú sa doň logické údaje popisujúce kroky zmien jednotlivých blokov, na rozdiel od ich celých obsahov. Je možné mať žurnál vedený interne, v rámci systému súborov spolu s dátami alebo na externom zariadení. XFS, podobne ako ext4 podporuje pozdržanú alokáciu (ako prvý súborový systém s touto funkciou), extenty premenlivej veľkosti (do 4 GB) a tiež priame vstupno-výstupné operácie. Taktiež pri alokácii blokov zohľadňuje samotný hardvér – počet diskov, čítacích a zapisovacích hláv; operácie sa snaží paralelizovať – využiť počet procesorov a pamäťových zariadení a zvyšovať priepustnosť toku dát. XFS je taktiež vhodnou voľbou pre dobrý súborový systém, najmä pre skúsených používateľov ochotných stráviť čas nastaveniami, ktoré poskytuje, s cieľom maximalizovať výkon. [Sch04, SGI]

Btrfs

Súborový systém Btrfs je jedným z najmladších – jeho počiatky siahajú iba do roku 2007 a ešte stále nie je vo finálnej, stabilnej verzii, no stojí za zmienku. Vyznačuje sa maximálnou podporovanou veľkosťou partície 16 EB a súbormi rovnakej veľkosti. Je to takzvaný copy-on-write systém, čo znamená, že pri čítaní sa volajúcim odovzdá iba smerník na údaje, ale reálna kópia sa vytvorí až pri prvom náznaku zápisu. Pri indexácií sú použité B+ stromy, odtiaľ plynie aj názov – B-tree filesystem, skrátene Btrfs. Medzi najdôležitejšie funkcie Brtfs (okrem už spomenutých) patrí:

- zmena veľkosti partície a defragmentácia za behu
- kompresia údajov
- podzväzky – sú to samostatné súborové systémy existujúce v rámci jednej partície pod Btrfs systémom. Každý podzväzok má názov a je možné ho nezávisle pripájať a odpájať (mount / unmount).
- snímky – predstavujú stav podzväzku – adresárov a súborov v danom časovom okamihu. Pomocou nich je možné jednoducho, automaticky a hlavne rýchlo vytvárať zálohy, pretože tvorba snímku je atomická operácia.
- kontrolné súčty dát a metadát

Veľkou výhodou Brtfs je tiež možnosť konverzie zo systémov súborov ext3 alebo ext4 a naspäť. Btrfs má veľký potenciál v použití vo veľkých systémoch, ako aj pri pracovných staniciach, kde je ho možné nasadiť už teraz a experimentovať s ním. Je pravdepodobné, že už v blízkej dobe bude vydaná stabilná verzia. Zatiaľ je podporovaný kernelom Linux-u vo verzii 2.6.29-rc1 a zahrnutý ako pokusný systém v niekoľkých väčších distribúciách. [Wikc]

2.2.2 Virtuálny systém súborov (VFS)

V predošlej časti boli uvedené súborové systémy na úrovni veľmi blízkej hardvéru. Fyzický disk, prípadne niekoľko diskov teda môžeme rozdeliť na niekoľko partícii a na každej spravovať nejaký typ podporovaného súborového systému, pričom tieto systémy môžu byť navzájom rozdielne. Aby bol však prístup k adresárom a súborom jednotný, ucelený, Linux všetky tieto partície a ich súborové systémy spája a vytvára nad nimi abstraktnú vrstvu nazvanú virtuálny systém súborov (angl. Virtual Filesystem – VFS). Pre používateľa sa dáta javia akoby boli na jednom médiu bez ohľadu na použité reálne systémy súborov. [Lov05, DPB05, Rus98]

Pre VFS sú dôležité štruktúry, s ktorými tento systém pracuje. Tieto sú:

- superblok
- inode – vrchol indexu (angl. index node)
- dentry – záznam pre adresár (angl. directory entry)
- súbor

Superblok

Táto štruktúra predstavuje základ každého pripojeného reálneho súborového systému a obsahuje jeho popisné údaje – metadáta. Medzi ne patrí napríklad celkový počet blokov v systéme, počet voľných blokov alebo číslo koreňového inode. Každý superblok je uložený na disku (kvôli redundancii, pre zvýšenie ochrany pri poškodení, na viacerých miestach) a tak poskytuje kernelu informácie o súborovom systéme, ktorý reprezentuje. Ak je nejaký reálny súborový systém pripojený do VFS, príslušný superblok je načítaný do pamäte. Z hľadiska vlastnej štruktúry obsahuje superblok atribút pre typ svojho súborového systému, pole pre zoznam všetkých inode, pole pre zoznam otvorených súborov, rôzne zámky a iné. Ďalej tiež metódy pre

ich obsluhu, napríklad pre načítavanie a zápis inode alebo znovapripojenie. [oL, Lov05, DPB05]

inode

Inode je v Linux-e základným prístupovým bodom k súboru, adresáru alebo linku. Existujú však rôzne typy inode. Reálny súborový systém má svoje vlastné, ktoré však nie sú súčasťou štruktúr VFS, pretože ten spravuje svoje. Líšia sa predovšetkým tým, že inode reálneho súborového systému je perzistentný, uložený na disku a udržiava údaje o objektoch špecifických pre daný súborový systém. Naproti tomu inode VFS predstavuje abstrakciu pre reálny inode a je uložený v pamäti. Podobne ako superblok, aj štruktúra pre inode má svoje atribúty obsahujúce metadáta popisujúce súbor a metódy pre manipuláciu so súborom, ktorý reprezentuje. Ide napríklad o čas vytvorenia a poslednej úpravy, vlastníka a jeho práva, či zoznam dentry, ktoré na inode smerujú. Niektoré metódy inode pre prácu so súborom sú priamo mapované na systémové volania, napríklad otvorenie, načítanie a zápis do súboru, no obsahuje aj metódy špecifické pre úroveň inode – vytvorenie, vyhľadanie, prelinkovanie a podobne. Ďalej obsahuje smerník na objekt, ktorý spravuje adresný priestor súboru – načítavanie stránok, mapovanie do adresného priestoru procesov. [oL, Lov05, DPB05]

dentry

Ďalšou štruktúrou vo VFS je dentry zastupujúci adresár. Každý dentry objekt má rodičovský dentry objekt, ktorý naň smeruje, okrem koreňového, ktorý je referencovaný superblokom. Atribútmi dentry sú názov adresára, smerník na superblok systému súborov, ktorému fyzicky prislúcha, smerník na rodičovské dentry nasledovaný zoznamom dentry, ktorým je dané dentry rodičom a tiež smerník na prislúchajúci inode. Metódami sú operácie s adresárom ako napríklad vytvorenie, zmazanie a načítanie obsahu. [oL,

Lov05, DPB05]

Súbor

V Linux-e, každý otvorený súbor je reprezentovaný objektom, ktorý obsahuje metadáta o súbore – všeobecné aj viazané na používateľa. Tento objekt má atribúty pre rôzne príznaky, práva pre používateľa a skupinu, smerník na rodičovské dentry, atribút pre mód, v akom je súbor otvorený a stavové atribúty ako napríklad aktuálna pozícia v súbore. Metódy pre túto štruktúru sú štandardné operácie so súborom – otvorenie, zatvorenie, čítanie, zápis a podobne.

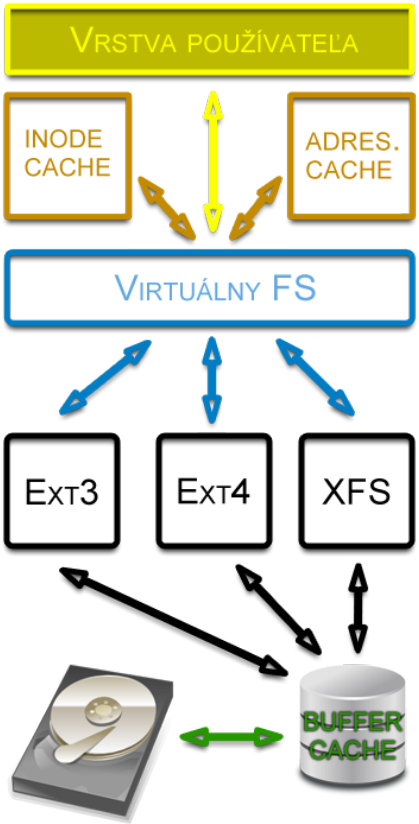
Ako však VFS funguje? Všetky súborové systémy, ktoré Linux podporuje, používajú takzvanú buffer cache. Je to vyrovnávacia pamäť blokových zariadení spoločná pre každý reálny súborový systém slúžiaca pre zrýchlenie načítavania a zápisu údajov z / na fyzické zariadenia a zároveň sa tak súborové systémy stávajú nezávislými od pamäťových zariadení a ich ovládačov. Všetky blokové zariadenia tak poskytujú jednotné rozhranie pre prácu s nimi. Ak chce nejaký súborový systém pracovať s dátami na pamäťovom zariadení, postará sa o to príslušný ovládač, ktorý tieto dáta načíta do buffer cache, kde sú už pre systém prístupné. Jednotlivé bloky sú identifikované číslom bloku a identifikátorom zariadenia, takže v prípde, že by boli znova potrebné, použijú sa bloky v buffer cache a celý proces sa tak zrýchli.

Pri bootovaní systému, ako sa sa inicializujú jednotlivé reálne súborové systémy, registrujú sa zároveň do VFS, pričom dochádza k načítaniu ich superbloku. Pri operácii načítania superbloku ďalej dôjde k vytvoreniu stromu inode a následne k namapovaniu superbloku reálneho súborového systému na príslušný superblok vo VFS.

Pri prístupe k súboru alebo adresáru dochádza k traverzovaniu stromu inode VFS, pričom pri návšteve daného inode sa tento presunie do takzvanej inode cache. Prístup do cache pamäte je totiž oveľa rýchlejší ako traverzo-

vanie cez inode v systéme súborov a teda pri opakovaných prístupoch k nejakému súboru, či adresáru, ďalšie už budú rýchlejšie; to platí aj pre súbory, či adresáre, ktoré sú v nejakom podstrome traverzovania a teda sú v cache. Z inode cache sú potom odstraňované málo pristupované inode. Podobne funguje adresárová cache pre adresáre. Tá však neobsahuje inode adresárov, ale len mapovanie názvov adresárov na príslušné čísla inode, ktoré sú v inode cache.

Prístup k súboru potom spočíva v rozdelení jeho plnej cesty a vyhľadaní adresárov. Ak sú tieto adresáre v adresárovej cache, systém pozná ich VFS superblok, ktorý je namapovaný na superblok reálneho súborového systému, ktorému je táto úloha ďalej delegovaná. V prípade, že adresár sa v adresárovej cache nenachádza, situácia je v zásade rovnaká, ale je potrebné začať prístup od koreňového adresára cesty k súboru, ktorý je priamo naviazaný na superblok. [oL, Sob05, Lov05, DPB05]



Obr. 2.1: Schéma komunikácie

Na obr. 2.1 je znázornená kominukácia jednotlivých častí systému súborov – pevný disk, buffer alebo bloková cache pamäť, reálne systémy súborov, virtuálny systém súborov, cache pre inode, cache pre adresáre a priestor používateľa – aplikácie.

2.2.3 Hierarchia systému súborov, FHS

VFS teda umožňuje jednotný pohľad na štruktúru adresárov a súborov, nezávislý od hardvéru a reálnych súborových systémov. Štruktúru adresárov a súborov nazývame hierarchiou systému súborov a Linux ju prevzal od Unix-u. V iných operačných systémoch nemusí byť adresárová štruktúra viazaná na význam jednotlivých adresárov alebo súborov; napríklad od aplikácií sa nepožaduje inštalácia do konkrétneho, stanoveného adresára a je v zásade na používateľovi aké umiestnenie zvolí. Takisto pre jeho súkromné dáta. Aplikácie taktiež zvyknú byť ucelené v zmysle enkapsulácie svojich spustiteľných súborov, konfiguračných súborov a dát do jedného adresára. V Linux-e je to však inak. Adresáre majú svoj význam, funkciu a rozhodnutie, kde sú jednotlivé súbory uložené je závislé od ich významu a vo veľkej miere pod kontrolou Linux-u a nie používateľa. Taká je základná filozofia. Pôsobí deterministicky a transparentne, no spočiatku, ako vznikali prvé distribúcie Linux-u, nastali problémy aj v tejto oblasti, pretože vydavatelia distribúcií si význam jednotlivých adresárov a celkovú výslednú štruktúru určovali sami. To bolo dôvodom vzniku štandardu, ktorý by hierarchiu zjednotil. Nazýva sa FHS (Filesystem Hierarchy Standard) a momentálne je vo verzii 2.3 (vydaná 29. 1. 2004). FHS sa teda skladá z množiny požiadaviek a direktív pre usporiadanie adresárovej a súborovej štruktúry za účelom interoperability aplikácií, nástrojov a skriptov pre všetky Linux-ové distribúcie, ktoré sa tohoto štandardu držia. FHS popisuje nasledovnú štruktúru:

- /

  Koreňový adresár celej štruktúry. Jeho úlohou je bootovať, obnovovať alebo opravovať celý operačný systém. Pre bootovanie je na primárnej partícii potrebný boot loader, konfiguračné súbory, utility. Partícia pre koreňový adresár sa odporúča čo najmenšia z viacerých dôvodov.

Funkčným dôvodom je, že takto môže byť základ systému prenositeľný na kapacitne obmedzenom médiu. Bezpečnostným, že ak sa vyskytne poškodenie partície, bude poškodených menej súborov a kompaktný systém je ľahšie udržiavaný a obnoviteľný. Posledným dôvodom môže byť zbytočnosť veľkej primárnej partície – jednoducho to nie je potrebné a uvoľní sa tak miesto na médiu pre súbory s iným, pre používateľa dôležitejším významom, ktoré môžu byť ťažšie zachrániteľné. Aplikáciám je zakázané meniť štruktúru koreňového adresáru.
V koreňovom adresári je nutné mať nasledovné adresáre alebo symbolické linky na ne: /bin, /boot, /dev, /etc, /lib, /media, /mnt, /opt, /sbin, /srv, /tmp, /usr, /var.

- /bin

  Obsahuje základné systémové programy – utility (alebo symbolické linky na ne), bez ktorých by sa používateľ, ani administrátor nezaobišiel a systém by bol dé facto neovládateľný. Tieto utility musia byť dostupné aj v prípade, kedy je pripojená iba primárna partícia s koreňovým adresárom. Tieto utility sú tiež požadované boot skriptami. V /bin adresári nie sú povolené žiadne podadresáre. Adresár /bin obsahuje aj utilitu alebo symbolický link **sh** predstavujúci program Bourne Shell, no ak tento nie je prítomný, **sh** musí byť linkom na iný shell program. Zopár príkladov programov, utilít, ktoré sa v tomto adresári nachádzajú:

  – /bin/cp – program pre kopírovanie súborov a adresárov.
  – /bin/kill – program pre ukončenie procesu.
  – /bin/ln – program pre vytvorenie linku medzi súbormi, či adresármi.
  – /bin/ls – program pre zobrazenie obsahu adresára.

  - /bin/mount – program pre pripojenie systému súborov.
  - /bin/sh – program pre spustenie shell-u.
  - /bin/su – program pre zmenu používateľa.

- /boot

  Adresár so súbormi zabezpečujúcimi bootovanie systému. Konfiguračné súbory pre bootovanie sú však uložené v /etc adresári; no nachádzajú sa tu všetky súbory a skripty použité respektíve vykonané predtým, ako kernel začne spúšťať programy v používateľskom móde. Obsah /boot adresáru vyzerá štandardne takto:

  - /boot/boot.0300 – záloha MBR (master boot record).
  - /boot/vmlinuz-<verzia>-<architektúra>– samotný kernel.
  - /boot/grub – podadresár obsahujúci konfiguračné súbory pre GRUB.
  - /boot/grub/device.map – mapovanie zariadení z /dev na zariadenia používane GRUB-om.
  - /boot/grub/grub.conf alebo /grub/menu.lst – hlavný konfiguračný súbor pre GRUB.

- /dev

  Ide o adresár so špeciálnymi súbormi predstavujúcimi prístupové body k zariadeniam. Nachádzajú sa tu prístupy k partíciám, diskovým a disketovým mechanikám, periférnym a sieťovým zariadeniam. Musí obsahovať tieto súbory – zariadenia:

  - /dev/null – pseudo zariadenie ignorujúce akýkoľvek vstup, čítanie z tohoto zariadenia vracia EOF – koniec súboru.

  - /dev/zero – pseudo zariadenie ignorujúce akýkoľvek vstup, čítanie z tohoto vracia žiadaný počet nulových byte-ov.
  - /dev/tty – zariadenie pre terminál.

- /etc

  Veľmi dôležitý adresár – obsahuje hierarchiu konfiguračných súborov systému a programov. Konfiguračný súbor je definovaný ako lokálny súbor zabezpečujúci vykonávanie programu podľa očakávaní, nastavení používateľa alebo niekoho iného. Je to statický súbor, nesmie byť binárny. Preto tento adresár štandardne neobsahuje žiadne binárne, ale iba textové súbory. Je odporúčané robiť jeho časté zálohy. Niektoré podadresáre a ich význam:

  - /etc/default – východzie nastavenia pre programy.
  - /etc/sysconfig – obsahuje všeobecné konfiguračné súbory systému a tiež konfiguráciu bootovania, skutočný obsah je závislý od distribúcie.
  - /etc/skel – obsahuje východzie konfiguračné súbory pre používateľa, ktoré sú po jeho vytvorení nakopírované do jeho domáceho adresára.
  - /etc/opt – obsahuje konfiguračné súbory programov špecifické pre danú inštaláciu.
  - /etc/X11 alebo /etc/kde – obsahuje konfiguračné súbory pre grafické používateľské prostredie špecifické pre danú inštaláciu.
  - /etc/ssh – obsahuje konfiguračné súbory pre Open Secure Shell protokol určený pre vzdialené prihlasovanie a obsluhu.
  - /etc/apache2 alebo /etc/httpd – obsahuje konfiguračné súbory pre HTTP server, zvyčajne Apache.

  - /etc/rc.d – v tomto adresári sú definované skripty pre jednotlivé úrovne behu (viac o úrovniach behu v časti 2.4.3 – Štart systému); názvy týchto skriptov majú spravidla špeciálny tvar a to S<poradové číslo><názov>pre skripty, ktoré štartujú nejaký proces alebo K<poradové číslo><názov>pre skripty, ktoré nejaký proces ukončujú. Poradové číslo skriptu určuje poradie jeho spustenia pri prechode na danú úroveň behu.

- /home

  Keďže Linux podporuje viacero používateľov, je potrebné niekde uchovávať ich osobné dáta, ktoré budú pred ostatnými chránené. Na to slúži práve adresár /home alebo presnejšie, jeho podadresáre. Tento adresár zvykne výrazne narastať a pre jeho lepšiu správu je vhodné mať vyhradenú samostatnú partíciu. Jeho podadresáre sú v tvare /home/<meno používateľa>a vo východiskovom štádiu obsahujú zväčša len údaje o konkrétnom používateľovi – napríklad históriu shell príkazov alebo nastavenia grafického prostredia. Ďalej je ich štruktúra pod kontrolou používateľa a nie je štandardizovaná.

- /lib

  V tomto adresári sa nachádzajú moduly kernelu a zdieľané knižnice (napríklad základná knižnica pre programovací jazyk C – libc) potrebné pre nabootovanie systému, tiež však pre základné utility v /bin a /sbin a iné programy. Moduly kernelu a iné dôležité systémové knižnice sú konkrétne v podadresári /lib/modules/<verzia>-<architektúra>; ovládače zariadení napríklad v /lib/modules/<verzia>-<architektúra>/drivers. Štandard povoľuje ešte adresáre v tvare /lib<niečo>pre rozlíšenie verzií knižničných súborov napríklad pre 32 a 64 bitovú architektúru.

- /media

  Tento adresár predstavuje prístupový bod pre pripojiteľné zariadenia ako sú disketová, CD, Zip mechanika. Voľakedy boli tieto zariadenia pripojené priamo pod koreňový adresár, ale v prípade ich veľkého počtu to nepôsobilo dobre a tak bol vytvorený samostatný adresár s touto funkciou.

- /mnt

  Adresár /mnt predstavuje prístupový bod pre pripojiteľné systémy súborov. Umožňuje tak systémovému administrátorovi dočasne pripojiť a zdieľať napríklad ďalší pevný disk. Štandard zakazuje používať tento adresár inštalačnými programami.

- /opt

  V tomto adresári by sa mali nachádzať prídavné balíčky a programy, ktoré nie sú súčasťou základnej inštalácie. Ide napríklad o kancelárske balíky, internetové prehliadače. Adresár /opt má vyhradené podadresáre /opt/bin, /opt/doc, /opt/include, /opt/info, /opt/lib a /opt/man pre systémového administrátora a programom je zakázané používať tieto adresáre pre svoje funkčné súbory.

- /proc

  Je to špeciálny adresár, a ako napríklad /dev, tiež obsahuje virtuálny systém súborov. Neobsahuje totiž žiadne skutočné súbory s reálnymi dátami uloženými na disku, ale súbory v ňom poskytujú informácie o

stave systému, pamäte, o procesoch, pripojených zariadeniach, hardvérovej konfigurácii a podobne. Je teda často používaný kernelom a systémovými utilitami pre kontrolu chodu systému.

- /root

  Je to voliteľný adresár určený ako /home adresár pre root používateľa. Podľa štandardu FHS nie je povinný, ale odporúčaný.

- /sbin

  V tomto adresári sa nachádzajú programy pre administráciu systému. Sú to napríklad nástroje pre bootovanie, obnovu a opravu systému, ktoré často využívaju utility v adresári /bin. Ak je pripojený adresár /usr, väčšina týchto nástrojov sa nachádza tam, no tie sú považované za menej významné (nie sú potrebné pre údržbu systému ako takého).

- /srv

  Tento adresár je určený pre aplikačné služby ponúkané konkrétnym systémom. Štandard FHS však neurčuje vnútornú štruktúru tohto adresára; táto časť ešte nie je dopracovaná a tak je odporúčané tento adresár mať, no nespoliehať sa na existenciu súborov v ňom.

- /tmp

  Obsahuje dočasné súbory programov. Vymazávanie súborov je povolené robiť špecificky, podľa potreby, no štandardom je odporúčané mazať obsah až pri bootovaní systému.

- /usr

  Je to v podstate systém súborov v systéme súborov. To, čo predstavovali predošlé adresáre pre systém, adresár /usr predstavuje pre používateľa, či používateľov. Obsahuje všetky jeho dodatočne inštalované programy, ich dokumentáciu, knižničné a hlavičkové súbory. Nachádzajú sa tu napríklad súbory pre grafické prostredie, programy pre komunikáciu (telnet, ftp klient a podobne). Jeho obsah by mal byť zdieľaný respektíve nezávisle zdieľateľný (v prípade pripojenia /usr k inému systému spĺňajúcemu kritériá FHS). Veľké softvérové balíčky musia mať vytvorený vlastný podadresár v /usr a nepoužívať podadresáre definované štandardom. Povinné podadresáre v tejto štruktúre:

  - /usr/bin – obsahuje binárne súbory, takmer všetky programy používateľa.
  - /usr/include – obsahuje hlavičkové súbory pre programy v jazyku C potrebné pre kompiláciu programov v požívateľskom móde.
  - /usr/lib – obsahuje knižnice.
  - /usr/local – obsahuje lokálne špecifické používateľské programy vzhľadom na konkrétny systém.
  - /usr/sbin – obsahuje systémové programy, nástroje, ktoré nie sú z hľadiska funkčnosti systému kritické.
  - /usr/share – obsahuje používateľské dáta so širokým spektrom použiteľnosti, nezávislé na architektúre.
  - /usr/share/man – obsahuje súbory manuálov programov v /usr.

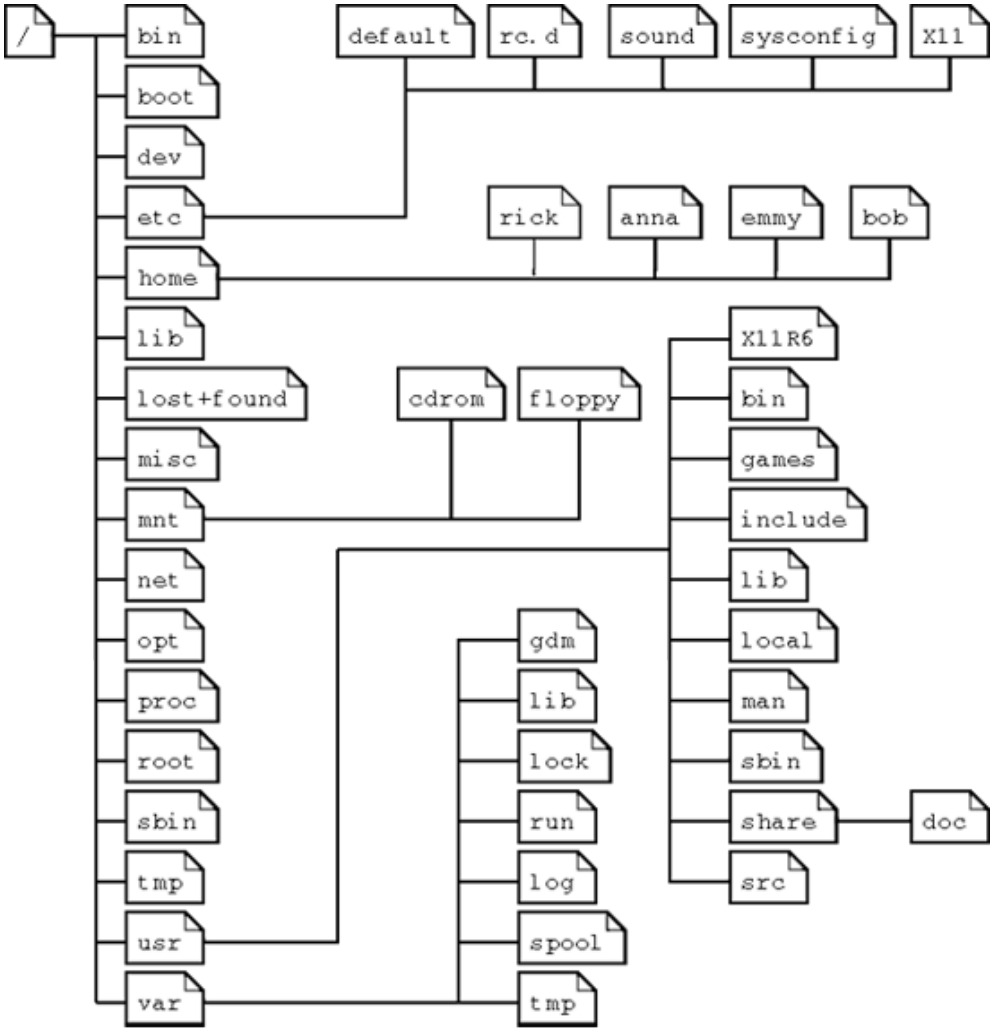
- /var

Tento adresár obsahuje premenlivé, zapisovateľné súbory využívané programami. Sú to napríklad logy, „spool“ súbory pre e-mail a tlač a tiež niektoré dočasné súbory. Niektoré podadresáre ako napríklad /var/log, /var/lock a /var/run nie sú zdieľateľné medzi rôznymi systémami. Dôvodom, prečo je táto časť oddelená od programov samotných, je možnosť mať výkonnú časť nezávislú a mať ju prípane na zariadení určenom len na čítanie. Je odporúčané vytvoriť vlastnú partíciu pre /var adresár. Štandard zakazuje vytvoriť link /var na /usr, pretože môže dôjsť ku konfliktom názvov podadresárov a došlo by k zlúčeniu významov týchto adresárov, čo je neželaný efekt. Je však možné vytvoriť link z /var na /usr/var. Programom je zakázané vytvárať podadresáre; tie by mali byť vytvorené len pokiaľ majú širší význam. Povinné podadresáre v tejto štruktúre:

- /var/cache – obsahuje cache pre programy, napríklad medzivýsledky algoritmov; vo všeobecnosti sú tieto dáta bezstratovo zmazateľné.
- /var/lib – obsahuje súbory s údajmi o aktuálnom stave programov, napríklad hodnoty premenných.
- /var/local – obsahuje dáta pre lokálne programy, ktoré by sa mali nachádzať v /usr/local.
- /var/lock – obsahuje zámky zariadení vyžiadané procesmi.
- /var/log – obsahuje súbory logov – informačných alebo iných záznamov programov a služieb.
- /var/opt – obsahuje dáta pre dodatočne inštalované programy v /opt.
- /var/run – obsahuje dáta spojené s informáciami o procesoch a aktuálnym stavom systému, napríklad informácie o práve prihlásených používateľoch.

- /var/spool – obsahuje fronty pre zariadenia a programy, ktoré ich vyžadujú.
- /var/tmp – obsahuje dočasné súbory väčšieho významu ako v /tmp, aby napríklad neboli zmazané pri bootovaní systému.

[Fou, Proa, Sob05, Sch04, Gar08, GOL]



Obr. 2.2: FHS

Na obr. 2.2 je znázornená štruktúra FHS aj s niektorými nepovinnými adresármi. [GOL]

2.2.4 Súbory

Existuje nasledujúce fundamentálne tvrdenie: „V Unix-ovom systéme je všetko súborom; ak to súborom nie je, je to proces.“ [GOL] Toto tvrdenie je pravdivé, napríklad aj všetky zariadenia sú reprezentované súborom a takisto aj všetko ostatné. Adresáre sú takisto súbormi, avšak obsahujúce ďalšie súbory. Keďže Linux je Unix-ový systém, toto tvrdenie sa vťahuje tiež naň.

Názvy súborov sú reťazce dĺžky najviac 255 ľubovoľných znakov s výnimkou zátvoriek a lomítka – /, to je vyhradené pre označenie koreňového adresára a ako oddeľovač adresárov v zápise cesty k súboru. Podľa konvencie sa však používajú iba alfanumerické znaky a špeciálne znaky sú obmedzené na množinu -, _, +, ., iné špeciálne znaky totiž môžu spôsobovať problémy v príkazoch shellu. Názvy súborov zvyčajne neobsahujú veľké písmená, no Linux ich rozlišuje, je to teda case-sensitive systém. Koncovky súborov majú čisto informatívny charakter, ináč súbor neovplyvňujú. V adresári nie je povolené mať viac súborov s rovnakým názvom. [Sob05, DPB05]

Typy súborov

V systéme súborov Linux-u sa samozrejme väčšinou nachádzajú regulárne, známe typy súborov aké sa nachádzajú v takmer všetkých operačných systémoch, no taktiež špeciálne.

- regulárne súbory – medzi ne patria napríklad spustiteľné, textové a mediálne súbory; jednoducho všetko iné ako ďalšie typy v zozname.
- adresáre (d) – už spomínané – sú to súbory obsahujúce súbory.
- znakové zariadenia (c) – predstavujú prístupový bod k znakovým zariadeniam a tak uľahčujú ich použitie.
- blokové zariadenia (b) – predstavujú prístupový bod k blokovým zariadeniam a tak uľahčujú ich použitie.

- doménové sokety (s) – majú podobnú funkcionalitu ako sieťové TCP/IP sokety, no slúžia pre sieťovú komunikáciu medzi procesmi a sú chránené prístupovými právami ako iné súbory.
- pomenované rúry (named pipes) (p) – tiež umožňujú komunikáciu medzi procesmi, ale mimo sieťových protokolov.
- symbolické linky (l) – súbory zastupujúce skutočný súbor – dáta v štruktúre súborov; vytvárajú tak akoby nové umiestnenie súboru v štruktúre, pričom nezaberajú kapacitu súboru ako keby bol súbor skutočne premiestnený.

[DPB05, Sch04, Gar08]

Práva

Nastavenie prístupových práv pre súbory je základnou úrovňou ochrany pred zneužitím údajov, či systému. Kritickými sú napríklad hlavné konfiguračné súbory, skripty pre firewall alebo vlastné súbory jednotlivých používateľov. Linux rozlišuje pre súbor štytri druhy používateľov:

- root
- vlastník súboru – zvyčajne to býva používateľ, ktorý súbor vytvoril.
- skupina – množina používateľov s rovnakými právami pre súbor; zvyčajne sa v nej nachádza aj vlastník súboru, no nie je to pravidlom.
- svet – reprezentuje ostatných používateľov.

A taktiež pre každého z nich rozlišuje tri druhy práv: právo pre čítanie, zápis a spustenie. Root má pre každý súbor rovnaké, plné práva. Pre zvyšných používateľov, pre zjednodušenie notácie, sú práva symbolicky reprezentované trojciferným kódom. Každá cifra predstavuje súčet čísel pridelených druhu práv pre jednotlivý druh používateľa. Čísla pridelené právam:

- 0 – žiadne práva
- 1 – právo na spustenie súboru
- 2 – právo na zápis do súboru
- 4 – právo na čítanie súboru

Existujú tiež špeciálne druhy práv – príznaky:

- „Sticky“ bit – v minulosti sa používal pre binárne spustiteľné súbory, programy, ako určenie zachovania kódu v pamäti aj po skončení vykonávania programu. Dôvodom bolo šetrenie diskových operácií, keďže disky neboli také rýchle, no v súčasnosti sa pre súbory nepoužíva. Zostáva ale pri adresároch – tu nastavenie „Sticky“ bit príznaku zamedzuje vymazanie alebo premenovanie súborov v danom adresári inými používateľmi ako vlastníkovi daného súboru alebo adresára.
- SUID bit – ak je binárny spustiteľný súbor označený týmto príznakom, po jeho spustení kýmkoľvek, kto má na jeho spustenie právo, sa nastaví vlastník príslušného procesu na hodnotu vlastníka súboru.
- SGID bit – podobne ako predošlý príznak, no uplatňuje sa skupina procesu a súboru.

[Sob05, DPB05, Sch04, Gar08, SG03]

2.3 Shell

Shell, nazývaný aj príkazový riadok, je súčasťou každej Linux-ovej distribúcie a predstavuje základné – textové komunikačné rozhranie medzi systémom a používateľom. Táto komunikácia spočíva v priamom zadávaní jednoduchých, no aj komplexných príkazov, ktoré shell interpretuje, využívajúcich utility a

aplikácie v systéme. Umožňuje tak používateľovi napríklad prehliadať systém súborov, spúšťať programy a podobne. Všetky aplikácie shellu podporujú interpretáciu regulárnych výrazov pre mená súborov, rúry, presmerovanie výstupu, premenné, podmienky a cykly. [Sob05, Gar08]

Najpoužívanejšie alebo najvýznamnejšie aplikácie pre shell:

- **bash** – Bourne-Again Shell
- **csh** – C Shell
- **tcsh** – Tenex C Shell
- **ksh** – Korn Shell

bash

Bourne-Again Shell je Linux-ová reedícia legendárneho Unix-ového shellu nazvanom podľa jeho tvorcu – Bourne Shell. Bol to základný, starý shell s veľmi obmedzenou funkcionalitou na dnešné pomery, no stal sa počiatočným krokom pre ďalší vývoj v tejto oblasti.

Bash bol vydaný pod hlavičkou GNU ako náhrada za proprietárny Bourne Shell a je štandardom pre Linux. Jeho funkcionalita je nadmnožinou Bourne Shell-u a tak všetko, čo fungovalo, zostalo zachované, ale prináša naviac nové možnosti, inšpirované hlavne shellmi ksh a csh. Jeho používanie je vcelku intuitívne a je flexibilný z hľadiska konfigurovateľnosti. Jednotlivé príkazy je možné spájať do súborov – skriptov vo forme algoritmov. [Sob05, Gar08, Wikb]

csh

C Shell – bol vcelku žiadaný programátormi, pretože jeho syntax je podobná ako syntax programovacieho jazyka C. Bol vyvinutý ešte pred Bourne Shell-om, určený pre BSD Unix, no oproti Bourne Shell-u poskytoval naviac naprík-

lad históriu príkazov a aliasy. Priniesol pokrok v prehľadnosti a celkovej jednoduchosti používania. V súčasnosti už však nie je príliš používaný. [Gar08, Wikd]

tcsh

To, čo bash predstavoval pre Bourne Shell, predstavuje tcsh pre csh. Ide teda o nadmnožinu pôvodnej funkcionality a poskytuje ešte väčší používateľský komfort a rýchlosť, napríklad automatické dopĺňanie názvov súboru pri písaní príkazu. V systéme FreeBSD tak tcsh nahradil csh ako východziu shell aplikáciu. [Sob05, Gar08, Wiko]

ksh

V počiatkoch to bol komerčný softvér vyvinutý pre AT&T, no v roku 2005 bol uvoľnený pre nelicencované používanie. Korn Shell vznikol spojením funkcionalít Bourne Shell-u a C Shellu-u, no priniesol tiež nové možnosti. Orientoval sa pre pohodlie, ale zároveň rozšírenie možností písania skriptov a tým zefektívnenie práce používateľov. V ďalších verziách bola doplnená podpora aj pre skriptovacie jazyky ako napríklad perl, tcl alebo awk a teda Korn Shell umožňuje spúšťať skripty napísané v týchto jazykoch. Stal sa veľmi populárnym a v 80-tych rokoch AT&T deklarovalo 80 percentný podieľ ksh u ich zákazníkov. [Gar08, Wikk]

2.4 Bootovanie a štart systému

Pre uvedenie PC alebo iného podobného zariadenia do prevádzky, schopnej vykonávať užitočnú činnosť, je vo všeobecnosti najprv potrebné spustiť operačný systém a vykonať súvisiace operácie. Tento proces, počnúc zapnutím počítača a končiac spustením prvého programu v používateľskom móde, nazý-

vame bootovanie alebo zavedenie operačného systému. Čo sa však v skutočnosti pri tomto procese deje?

Po spustení prebehne takzvaná POST (Power-On Self Test) kontrola, pri ktorej ide o preskúšanie dostupnosti a základných funkcií potrebného hardvéru. Túto fázu vykonáva BIOS (Basic Input-Output System) a je nezávislá od operačného systému. Po tejto základnej kontrole BIOS inicializuje hardvér a podľa nastavení a dostupnosti lokalizuje zariadenie pre nabootovanie systému. Tým býva zvyčajne primárny pevný disk. Jeho prvý sektor (512 B) je označovaný ako MBR (Master Boot Record), teda hlavný záznam pre zavedenie systému a obsahuje základný kód pre ďalšie kroky. BIOS, nazývaný primárny boot loader, po tom, ako obsah MBR nahrá do pamäte, odovzdá mu riadenie.

Keďže primárny boot loader je výrazne priestorovo obmedzený, jeho úlohou je len vyhľadať aktívnu partíciu a načítať jej vlastný PBR (Partition Boot Record), obsahujúci sekundárny boot loader, do pamäte a vykonať ho. Sekundárny boot loader má už väčšiu zodpovednosť a to konkrétne načítanie skomprimovaného obrazu kernelu systém. V hlavičke tohto obrazu sa nachádza rutina vykonávajúca základné nastavenie hardvéru a následnú dekomprimáciu kernelu. Táto rutina potom odovzdá riadenie kernelu.

Kernel spočiatku inicializuje niektoré potrebné dátové štruktúry (napríklad zásobník), potom zavolá funkciu `start_kernel()` (/init/main.c), hlavnú funkciu celého kernelu. Tá vykoná ďalšie inicializačné rutiny – pripravenie prerušení, konfigurácia pamäte. V tejto fáze sa tiež môže do pamäte načítať a pripojiť počiatočný RAM disk (initrd). Ten slúži ako dočasný systém súborov a umožňuje kernelu bootovanie bez potreby pripojenia fyzických pevných diskov. Initrd tiež môže obsahovať dynamické moduly pre periférne zariadenia a tak zachovať kernel veľmi malý, obsahujúci len to najnutnejšie. Po nabootovaní kernelu sa initrd odpojí a pripojí sa skutočný koreňový systém súborov a ostatné subsystémy definované v konfiguračnom súbore /etc/fstab.

V závere sa zavolá program `init` (zvyčajne /sbin/init), už v používateľskom priestore. [Fle05, War04, Sch04]

2.4.1 LILO

Prvým používaným boot loaderom bol Linux Loader alebo v skratke LILO. Pomocou neho bolo možné nabootovať aj iné operačné systémy v závislosti od obsahu partícií, ale v súčasnosti je zastaralý a nie úplne vyhovuje podmienkam moderných systémov, nepodporuje napríklad iné súborové systémy ako ext2, či ext3. Umožňuje však po svojom spustení zobraziť výber kernelov, ktoré je možné nabootovať. [Fle05, War04, Sch04]

2.4.2 GRUB

GRUB, alebo GRand Unified Bootloader je vyspelejším nasledovníkom LILO boot loaderu. Hlavnou výhodou oproti LILO je znalosť typov súborových systémov a možnosť takto bootovať kernel z viacerých typov ako len z ext2. GRUB tiež dokáže zobraziť zoznam kernelov, ktoré je možné bootovať – používa pri tom konfiguračné súbory /etc/grub.conf a /etc/grub/menu.lst. [War04]

2.4.3 Štart systému

Samotný štart programov je, ako bolo spomenuté, zahájený spustením programu `init`. Ten načíta svoj konfiguračný súbor – /etc/inittab a riadi sa ním. V tomto súbore sú nadefinované akcie, ktoré má systém vykonať:

- sysinit – reprezentuje udalosť ešte pred prepnutím systému do nejakej úrovne behu.
- úrovne behu (runlevels) – reprezentujú akoby stav systému, každá úroveň

behu má totiž nadefinované aké programy sa majú spustiť alebo zastaviť, jednotlivé štandardné úrovne behu sú:

  - 0 – zastavenie systému (halt).
  - 1 – jednopoužívateľský mód (single-user mode).
  - 2 – mód s viacerými používateľmi – prihlásením (multi-user mode).
  - 3 – mód s viacerými používateľmi – prihlásením a sieťovaním (multi-user mode with networking) – štandardne východzí mód.
  - 4 – používateľom definovateľná úroveň behu.
  - 5 – mód s viacerými používateľmi – prihlásením, sieťovaním a grafickým rozhraním (multi-user GUI mode with networking).
  - 6 – reštart systému (reboot).

- ctrlaltdel – udalosť, ktorá nastane, ak používateľ stlačí klávesy ctrl + alt + delete.

- respawn – udalosť, ktorá sa má zopakovať, ak pri nej definovaný príkaz, respektíve proces skončil svoju činnosť.

Týchto akcií je však viacej, no uvedené su najbežnejšie používané. V tomto konfiguračnom súbore je tiež možné nastaviť východziu úroveň behu, s ktorou bude systém spustený. Jednotlivé konkrétne skripty, ktoré sa majú spustiť pri danej úrovni behu sú ďalej definované v adresárovej štruktúre /etc/rc.d, o ktorej informuje časť 2.2.3. [Fle05, War04, Sch04]

2.5 Programy a balíčky

Operačný systém by bol bez aplikačnej vrstvy zbytočný. V Linux-e sú aplikácie, programy, riešené formou balíčkov, teda akýchsi enkapsulácií súborov.

Vo väčšine prípadov takáto enkapsulácia zahŕňa údaje o balíčku, skompilované súbory programu, inštalačné (`makefile`) a konfiguračné súbory, dátové súbory (médiá a podobne), manuál a potrebné knižnice. Tieto balíčky bývajú spravované balíčkovacím systémom, čo je akýsi repozitár poskytujúci rozhranie pre manipuláciu s nimi. Takýto balíčkovací systém má niekoľko výhod:

- jednoduchá inštalácia a aktualizácia – za predpokladu, že je balíčkovací systém správne nakonfigurovaný, používateľ vie, ktorý balíček je potrebné nainštalovať a nenastanú neočakávané komplikácie; inštalácia, či aktualizácia balíčka spočíva buď v zadaní jednoduchého príkazu alebo v prípade grafického prostredia, v pár klikoch.
- vyhodnocovanie závislostí balíčkov – balíčky väčšinou majú vzájomné závislosti, teda využívajú funkcionalitu jeden druhého. Tieto závislosti by mali byť, v súlade so systémom, definované v balíčku a tak je možné jednoducho tieto závislosti vyhodnotiť a podľa toho potrebné balíčky automaticky doinštalovať.
- možnosť rôznych zdrojov – balíčky je možné inštalovať z offline (CD / DVD, pevný disk) aj online (internet, sieť) zdrojov – tie sú najbežnejšie používané. Všetky zdroje je možné nakonfigurovať; napríklad pri online zdrojoch je potrebné nadefinovať lokality repozitárov, odkiaľ sa budú balíčky sťahovať.
- rýchle vyhľadávanie balíčkov – keďže balíčky sú centralizované do repozitárov, vyhľadanie balíčka je jednoduché; repozitáre totiž ponúkajú pre túto a podobné žiadosti funkcie.
- ochrana autora a používateľa – balíčky je väčšinou možné podpísať autorovým tajným kľúčom a pri stiahnutí balíčka podpis overiť jeho verejným kľúčom, čím sa zabráni podvrhnutiu falošného balíčka.

[Sch04, Gar08, MKD05]

Rozlišujeme dva základné typy balíčkovacích systémov:

- RPM
- Debian

2.5.1 RPM

RPM je v distribúciách asi najrozšírenejší. Názvy balíčkov majú tvar <názov>-<verzia>-<vydanie>.<cieľová architektúra>.rpm. Cieľová architektúra môže byť určená špecificky alebo označením śrc re balíček so zdrojovým kódom, ktorý je určený pre kompiláciu na cieľovej architektúre, ńosrc re súbory pre zostavenie balíčka, či ńoarch re balíčky nezávislé od cieľovej architektúry. Balíček sa skladá zo štyroch častí:

- úvod – obsahuje základné informácie o balíčku, slúžiace najmä na identifikáciu balíčku v rámci balíčkovacieho systému. Ide o verziu RPM, typ balíčka, cieľovú architektúru, názov, určenie operačného systému, pre ktorý je balíček vytvorený a typ signatúry.
- signatúra – obsahuje informácie pre kontrolu integrity – hash balíčka a prípadne podpis autora pre kontrolu autenticity; hash a podpis sa tvoria z hlavičky a samotného archívu
- hlavička – obsahuje ďalšie informácie o balíčku slúžiace hlavne používateľovi – stručný popis, podrobný popis a podobne
- archív – obsahuje archív balíčka a teda súbory, ktoré sa inštalujú

Nástrojov pre manipuláciu s RPM balíčkami je niekoľko. Základným je `rpm`, ktorý zvláda zistenie informácií o balíčku (aj závislosti), inštaláciu, odstránenie, aktualizáciu, kontrolu integrity a autenticity a ďalšie pomocné operácie. Nedokáže však vykonať automatické vyhodnotenie závislostí

a prehliadanie repozitárov. Iné programy to však zvládajú – ide napríklad o `yum`, `zypper` a `apt-rpm`. Tieto sú obvykle používateľsky priateľské a ich ovládanie je intuitívne. [Sch04, Gar08, MKD05]

2.5.2 Debian

Tento systém nie je natoľko používaný čo do počtu distribúcií, ale distribúcie, ktoré ho používajú, patria medzi najrozšírenejšie – Debian a Ubuntu. Balíčky sú tvorené archívom skladajúcim sa z troch súborov:

- debian-binary – obsahuje verziu formátu balíčka
- control.tar.gz – obsahuje metadáta balíčka, dodatočné skripty pre inštaláciu, či odstránenie balíčka, súbor so zoznamom všetkých konfiguračných súborov v balíčku a hash a podpis pre kontrolu integrity a autenticity
- data.tar(.gz|.bz2|.lzma) – obsahuje vlastné dáta balíčka (komprimované podľa poslednej koncovky)

Názvy balíčkov majú tvar <názov>_<verzia>-<verzia debianu>_<cieľová architektúra>.deb, podobne ako pri systéme RPM. Metadáta balíčka obsahujú informácie o verzii, závislostiach a tiež o priorite balíčku, čo je informácia pre operačný systém, ako má alebo môže s balíčkom operovať.

Pre manipuláciu s Debian balíčkami sa používajú programy – základný `dpkg` alebo pohodlnejšie, s rozšírenou funkcionalitou `apt`, `Aptitude`, `Synaptic` alebo `KPackage`. Ich funkcionalita je podobná ako pri systéme RPM.

2.5.3 Manuálna inštalácia

Ďalšou možnosťou je priama, manuálna inštalácia balíčkov, respektíve archívov. K dispozícii môže byť buď archív s binárnymi – skompilovanými súbormi

ako bolo uvedené v úvode tejto časti, alebo zdrojovými súbormi, ktorý je potrebné skompilovať. To má niekoľko výhod, napríklad možnosť detailnej konfigurácie, teda poskytnutie úplnej kontroly nad výslednou aplikáciou v zmysle jej možností, obmedzenie závislostí na potrebné minimum a zároveň je tak program kompilovaný priamo pre cieľovú architektúru. Nevýhodami sú potreba vyhodnotenia a vyriešenia závislostí manuálne a potenciálna pracnosť, či zdĺhavosť kompilácie. [Sch04, Gar08, MKD05]

Kapitola 3

Tvorba distribúcie

V predchádzajúcich kapitolách sme si predstavili jednotlivé výsledné komponenty procesu tvorby distribúcie a spôsob akým tieto komponenty navzájom kooperujú. V tejto kapitole ukážeme, ako sa tieto komponenty vytvárajú a tiež ako sa vytvára inštalácia výslednej distribúcie.

Linux-ová distribúcia je vo všeobecnosti tvorená kernelom a softvérom, ktorý ju určuje – sú to aplikácie, knižnice, dokumentácia a podobne. Tento softvér je spravidla k dispozícii vo forme balíčkov, čo sú archívy obsahujúce všetky relevantné súbory. Napríklad pre programy to bývajú binárne súbory alebo zdrojový kód spolu s dokumentáciou a konfiguračnými, či inštalačnými skriptami. Dokumentácia okrem používateľskej príručky zväčsa obsahuje aj manuál ako daný program nainštalovať a referencie na iné balíčky, od ktorých je daný balíček závislý.

Distribúcie je možné tvoriť rôznymi spôsobmi, napríklad mnoho existujúcich distribúcií obsahuje nástroje, ktoré umožňujú vytvoriť vlastnú distribúciu, pričom ako základ výslednej distribúcie je použitá daná hostiteľská distribúcia. V nasledujúcich krokoch však budeme predpokladať, že distribúciu tvoríme „na zelenej lúke“, aby sme tak výsledný systém spravili bez väzieb na nejakú bežne používanú distribúciu a zároveň si osvojili princípy jej tvorby. Nepoužívame teda žiadnu aplikáciu, ktorá by tvorbu uľahčovala a hostiteľská

distribúcia bude predstavovať len prostredie, v ktorom je možné vykonať žiadané postupy.

Distribúciu je takto možné tvoriť dvoma spôsobmi:

- Výberom balíčkov so zdrojovým kódom, ktoré má distribúcia obsahovať a ich kompiláciou.
- Výberom už skompilovaných balíčkov – aplikácií, ktoré má distribúcia obsahovať a skopírovaním súborov, ktoré tieto balíčky obsahujú, do cieľovej adresárovej štruktúry novej distribúcie.

Tieto metódy je však možné pre kombinovať – niektoré balíčky skompilovať, pričom iné skopírovať. Výhodou kompilácie balíčkov je úplná kontrola nad týmto procesom a určenie možností pre kompiláciu, ktoré nemuseli byť v predkompilovanom balíčku zahrnuté. Kompilácia balíčka tiež vytvorí adresárovú štruktúru, ktorú tak netreba vytvárať ručne. Nevýhodou je však často nemalá časová a kapacitná náročnosť. Skopírovanie súborov skompilovaného balíčka je rýchle, no je potrebné určiť množinu súborov, ktoré sa majú skopírovať, čo nemusí byť jednoduché hlavne pre väčšie balíčky. Pri tejto metóde je tiež potrebné manuálne určiť cieľové adresáre jednotlivých súborov, pričom je potrebné využiť štandard FHS popísaný v časti 2.2.3. Tieto negatíva je však možné eliminovať predkompiláciou balíčka do zvláštneho adresára, čo spôsobí automatické vytvorenie adresárovej štruktúry a zároveň určí množinu súborov, ktoré treba skopírovať do cieľovej distribúcie.

Jednotlivé časti procesu tvorby je možné rozdeliť do nasledovných fáz:

- Príprava prostredia a základných elementov
- Kompilácia balíčkov a konfigurácia systému
- Inštalácia a štart systému

3.1 Príprava prostredia a základných elementov

Pre vytvorenie vlastnej distribúcie je potrebné mať nainštalovanú a spustenú už existujúcu distribúciu linuxu, pričom toto prostredie budeme nazývať hostiteľský systém. Hostiteľský systém poskytuje knižnice a aplikácie potrebné v prvých fázach tvorby vlastnej distribúcie, ak túto tvoríme zo zdrojových kódov balíčkov. Ak ju tvoríme vytvorením adresárovej štruktúry a skopírovaním existujúcich skompilovaných a rozbalených balíčkov, hostiteľský systém vytvára prostredie, v ktorom tento proces skladania môžeme vykonať.

Distribúciu môžeme vytvárať v novom adresári napríklad v adresárovej štuktúre adresára /home, či /tmp, no je odporúčané vytvoriť si pre tieto účely dedikovanú partíciu. Hlavným dôvodom pre vytvorenie partície je izolácia nového systému od hostiteľského na vyššej úrovni, ako keby bol systém tvorený v podadresári. Pri chybe v tvorbe, či už je to kopírovanie nejakej adresárovej štruktúry alebo kompilácia balíčka, je týmto zabezpečená menšia pravdepodobnosť, že dôjde k negatívnemu ovplyvneniu hostiteľského systému.

Ďalej je potrebné vymedziť balíčky, ktoré má distribúcia obsahovať. Tento krok je esenciálny pre tvorbu novej distribúcie, pretože určuje jej charakter. Ako bolo spomenuté v úvode kapitoly, balíčky majú často medzi sebou väzby, dependencie a teda použitie balíčka môže byť závislé od iných balíčkov, ktoré pre jeho správnu funkcionalitu musia byť vo výslednom systéme obsiahnuté. Rôzne matematické a systémové knižnice sú často vyžadované inými balíčkami, preto je do cieľovej distribúcie žiadané tieto knižnice zahrnúť. Balíčky je potrebné zaobstarať – zvyčajne je to možné na webovej stránke tvorcu balíčka, ďalšou možnosťou je získať ich skompilované z hostiteľského systému, ak v ňom existujú, no táto voľba nie je príliš vhodná, pretože súbory balíčka môžu byť príliš viazané na hostiteľský systém.

Pre tvorbu novej distribúcie je tiež odporúčané vytvoriť nového používateľa pre zamedzenie konfliktom s už vytvorenými používateľmi a je vhodné mu nastaviť systémové premenné, napríklad cestu do koreňového adresára novej distribúcie, pre uľahčenie procesu skladania.

V koreňovom adresári novej distribúcie je potom potrebné vytvorenie základnej adresárovej štruktúry podľa FHS, časť 2.2.3. Pri metóde kompilácie balíčkov sa však stačí obmedziť na adresáre /dev a /proc, pretože ostatné adresáre budú vytvorené automaticky procesom kompilácie jednotlivých balíčkov. Po vytvorení sú prázdne a pre úplnosť systému pri skladaní je do nich dočasne, kým systém nie je hotový, potrebné pripojiť adresárovú štruktúru týchto adresárov v hostiteľskom systéme. [Bee10]

3.2 Kompilácia balíčkov a konfigurácia systému

3.2.1 Kompilácia

Predpokladajme, že distribúciu budeme skladať spôsobom kompilácie balíčkov. To nám zabezpečí „čistý“ nový systém, ktorý bude plne izolovaný od hostiteľského. V prvých krokoch je potrebné vytvoriť si sadu kompilačných nástrojov. Je odporúčané použiť sadu GNU Toolchain, pretože táto je dobre otestovaná, stabilná a často využívaná. Tento krok je však tiež možné rozdeliť do dvoch fáz. V prvej zabezpečiť len nutnú množinu nástrojov, kompilovanú hostiteľským systémom, s pomocou ktorej je možné dokompilovať kompletnú sadu GNU Toolchain už v prostredí novej distribúcie.

Programy pre prvú fázu, ktoré sa skompilujú do špeciálneho oddeleného adresára v rámci novej distribúcie:

- GNU Binutils – základné nástroje pre manipuláciu s binárnymi súbormi, obsahuje programy:

  - as – assembler pre preklad jazyka symbolických inštrukcií do strojového kódu.
  - ld – linker pre spojenie objektov do spustiteľného binárneho súboru.
  - ďalšie nástroje pre prácu napríklad so symbolickými inštrukciami, objektovými súbormi.

- GNU C Library – štandardná knižnica, respektíve súbor knižníc programovacieho jazyka C; obsahuje funkcie pre základné systémové volania, matematické knižnice, knižnice pre prácu s reťazcami a inými dátovými typmi.

- GNU Compiler Collection – kompilátor projektu GNU s podporou viacerých programovacích jazykov, primárne je však určený pre jazyky C a C++.

- make – základný program pre automatickú kompiláciu programov a knižníc, ktorý pracuje na základe vopred vytvoreného popisného skriptu `makefile`.

Do prvej fázy je ešte potrebné zaradiť balíčky pre prácu s textovými reťazcami a súbormi, komprimačné, dekomprimačné programy, shell a podobne. Tieto sú však závislé od výberu balíčkov inštalovaných v druhej fáze, teda od toho aký systém bude v konečnom dôsledku vytvorený.

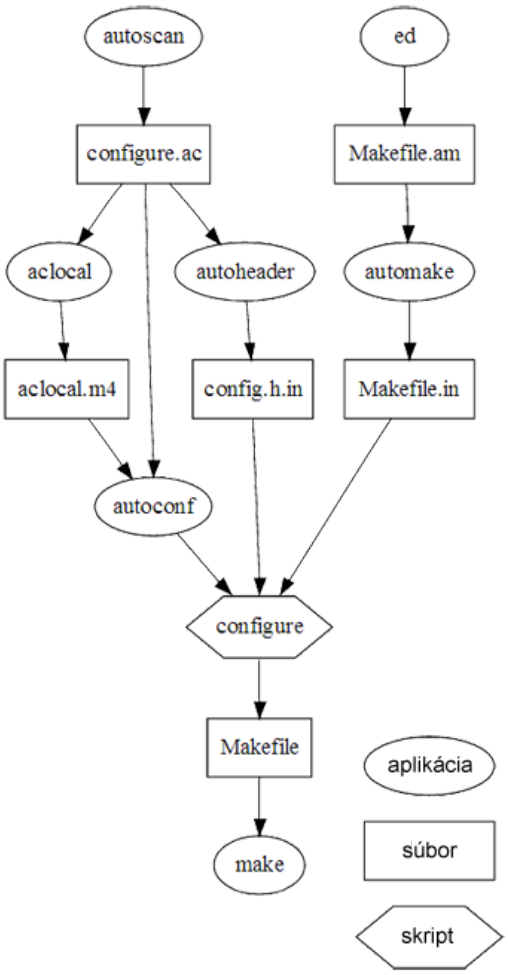
Po skompilovaní všetkých balíčkov potrebných pre autonómnu (v zmysle (ne)potreby použitia knižníc a nástrojov hostiteľského systému) kompiláciu výslednej distribúcie je potrebné prejsť do jej koreňového adresára príkazom `chroot`. Tento príkaz prenesie všetky ďalšie kroky a s nimi súvisiace väzby do prostredia novej distribúcie a zvyšok procesu tvorby tak bude od hostiteľského systému nezávislý.

Ako úvodný krok prvej fázy je potrebné najprv prekompilovať prvú časť GNU Toolchain (Binutils, glibc, GCC) do koreňového adresára novej dis-

tribúcie. V prvej fáze bola totiž táto časť iba dočasná, slúžiaca ako minimum pre prekompilovanie kompletnej sady GNU Toolchain, ktorú budeme ďalej využívať pre stavbu zvyšku celej distribúcie. V ďalšom kroku skompilujeme aj zvyšok GNU Toolchain:

- GNU Bison – generátor parserov; ako vstup spracuje definíciu LALR gramatiky, pričom výstupom je program, ktorý kontroluje syntaktickú správnosť zdrojových kódov vzhľadom na definovanú gramatiku.
- m4 – makroprocesor, vyžadovaný balíčkom Autoconf.
- GNU Autotools
  - Autoconf – spracováva vstupné konfiguračné súbory pre vytvorenie `configure` skriptu, ktorý ďalej slúži skriptu `makefile` (napríklad určuje knižnice dostupné v systéme, ktoré daný program vyžaduje).
  - Automake – slúži pre vytváranie `makefile` skriptov, ktoré definujú kompilácie balíčkov.
  - Libtool – nástroj pre vytváranie knižníc, zjednotenie a zjednodušenie ich používania; vo všeobecnosti však nie je nutný, no môže byť užitočný.

[Wikj, Bee10]



Obr. 3.1: Kompilácia

Na obr. 3.1 je znázornený proces kompilácie aj na strane vytvárania `configure` a `makefile` skriptov. Zdroj: [Wika])

Nová distrubúcia ale potrebuje aj ďalšie balíčky aby mohla byť využiteľná. Medzi ne patrí program pre shell, mnohé systémové programy (pre správu procesov, systémov súborov, zariadení), knižnice, boot loader. V neposlednom rade je to kernel, ktorý je takisto obsahom balíčku. Zvyšné balíčky už sú samotnou záležitosťou tvorcu distribúcie a ich výber je podriadený jej účelom. Avšak, ako bolo spomenuté v podkapitole 2.5 o balíčkoch, balíčky majú medzi sebou dependencie a tak pri zostavovaní aplikácií systému – ich výbere, je potrebné na tento fakt dbať ohľad.

3.2.2 Konfigurácia

V sekcii 2.2.3 je popísaná hierarchia systému súborov v Linux-e, pričom časť je venovaná štandardnému umiestneniu konfiguračných súborov. Tieto konfiguračné súbory je možné vytvoriť manuálne, ale rýchlejšia cesta je stiahnuť existujúci balíček, ktorý ich obsahuje. Niektoré konfiguračné súbory už boli spomenuté v iných častiach, no v tejto časti si hlavné konfiguračné súbory, ktoré by mali byť spoločné pre všetky Linux-ové distribúcie, zhrnieme a vysvetlíme na čo slúžia.

- bashrc – všeobecný konfiguračný súbor pre bash. Obsahuje globálne funkcie, premenné a nastavenia. Tento konfiguračný súbor sa vo všeobecnosti v každej distribúcii vyskytovať nemusí, no bash je natoľko rozšírená aplikácia shell-u, že je vhodné ho spomenúť.
- fstab – obsahuje zoznam systémov súborov, ich body pripojenia a súborové systémy na nich.
- group – zoznam skupín používateľov.
- hosts – zoznam IP adries a ich domén, ktoré sú prístupné cez sieť – pre priame mapovanie IP adresy a domény bez použitia DHCP.
- inittab – hlavný konfiguračný súbor použitý pri štarte systému.

- issue – informácie o distribúcii.
- ld.so.conf – obsahuje cesty, kde sa nachádzajú dynamické knižnice.
- modules – zoznam modulov, ktoré sa nahrajú do kernelu pri štarte.
- motd – správa zobrazená po prihlásení sa do systému, nie je však povinným súborom v distribúciách.
- mtab – obsahuje zoznam aktuálne pripojených systémov súborov.
- nsswitch.conf – poradie volania služieb pre prístup k informáciám o systéme.
- pam.d – adresár; obsahuje konfiguráciu pre autentifikáciu.
- passwd – zoznam používateľov v systéme.
- profile – najzákladnejší konfiguračný súbor pre globálne funkcie a premenné v shell-i (nie len pre bash).
- resolv.conf – poradie volania DNS serverov.
- services – zoznam sieťových služieb, protokolov a ich portov, ktoré daný systém poskytuje.
- shadow – obsahuje šifrované heslá používateľov a ich platnosť.
- shells – zoznam dostupných shell aplikácii v systéme.
- timezone – časová zóna.

[Gar08]

3.3 Inštalácia a štart systému

Existujú v zásade dve možnosti ako používať operačný systém – buď ako živú (live) distribúciu, kedy je systém nabootovaný z média (vo väčšine prípadov z CD / DVD nosiča) a nie je potrebná inštalácia, alebo máme systém nainštalovaný na pevnom disku. Prvou možnosťou sa však zaoberať nebudeme; nie je cieľom tejto práce a jej použitie je tiež značne špecifické vzhľadom na charakter použitého média.

Budeme teda uvažovať situáciu, v ktorej je našou intenciou nejakým spôsobom distribúciu nainštalovať na cieľový hardvér. To je možné napríklad pomocou inštalačného média, ktoré obsahuje všetky potrebné súbory alebo kombinovane (tento proces sa nazýva sieťová inštalácia) – inicializovať inštaláciu pomocou média, no inštalačné balíky získať pomocou siete. Pri sieťovej inštalácii bývajú súbory uložené na vzdialenom serveri, na cieľový hardvér sú stiahnuté pomocou sieťového protokolu a nainštalované sú základnými nástrojmi nachádzajúcimi sa na médiu. Tento spôsob umožňuje znížené kapacitné nároky na médium, ale inštalačný proces je zvyčajne pomalší.

Pri oboch typoch inštalácie je však potrebné najprv systém nabootovať. V minulosti sa ako bootovacie médium používala disketa, no v súčastnosti sú disketové mechaniky na ústupe a je preto stačí uvažovať len CD / DVD nosiče, prípadne USB kľúče. Pre tieto médiá je potrebné vytvoriť ISO obraz, no tento musí byť tvorený špeciálne s možnosťou bootovania. Pre tieto účely je vhodné použiť programy `mkisofs` alebo `genisoimage`, ktoré priamo ponúkajú možnosť vytvorenia takéhoto obrazu. Pre bootovateľný obraz tiež treba nastaviť boot loader, podobne ako je to pre nabootovanie už nainštalovaného systému v časti 2.4, no pre ISO obrazy sa okrem LILO a GRUB boot loaderu zvykne používať tiež ISOLINUX boot loader.

Boot loaderu na ISO obraze je potrebné nastaviť možnosti pri bootovaní – počiatočný kernel a RAM disk. Tieto predstavujú dočasný operačný systém,

pomocou ktorého je možné vytvoriť a pripojiť logické disky a nainštalovať na ne potrebné údaje. Inštalácia môže predstavovať skopírovanie hotového skompilovaného FHS, tvoreného v predošlej časti alebo kompiláciu balíčkov priamo na mieste, v cieli inštalácie. Väčšina existujúcich distribúcii využíva prvú možnosť. Dôvodom sú oveľa menšie časové nároky, pretože kompilácia zvyčajne vyžaduje netriviálny čas (pre distribúcie s desiatkami balíčkov to môže byťvyše desať hodín). Výhodou druhej možnosti je akási optimalizácia výkonu inštalovaných komponentov, keďže tento sa kompiluje priamo na cieľovom hardvéri.

Po nainštalovaní je potrebné uvoľniť bootovacie médium, spustiť reboot systému a prebehnú akcie deklarované v časti 2.4, už s použitím kernelu a ramdisku novej nainštalovanej distribúcie.

Kapitola 4

Generovanie distribúcie

Táto kapitola nadväzuje na predchádzajúcu kapitolu pojednávajúcu o manuálnej tvorbe distribúcie. Ukážeme, že tento proces sa dá zautomatizovať a je možné vytvoriť mechanizmy, ktoré ho uľahčujú do takej miery, že tvorba distribúcie je možná aj bez bohatých skúseností s Linux-om ako takým.

Pre automatizovanú tvorbu distribúcie existujú dva prístupy – offline a online generovanie. Offline nástroje sú reprezentované aplikáciami v rámci distribúcií, ktoré môžu slúžiť aj ako nástroje pre zálohovanie, ale zvyčajne ponúkajú aj možnosť tvorby vlastnej distribúcie založenej na hostiteľskom systéme – distribúcii, pre ktorú sú určené. Analýza týchto aplikácií je však nad rámec tejto práce a tak len uvedieme niekoľko z nich:

- Remastersys – aplikácia určená pre zálohovanie a tvorbu Debian-ovských distribúcií. [Bri]
- Live-Magic – ďalšia aplikácia určená pre Debian-ovské distribúcie. [Lam]
- Revisor – aplikácia určená pre tvorbu distribúcií založených na distribúcii Fedora. [Prob]

Relatívne novú triedu však tvoria online nástroje, ktoré rozoberieme v nasledujúcej časti.

4.1 Online nástroje pre generovanie distribúcie

4.1.1 Instalinux

Instalinux je online aplikácia pre zostavovanie inštalácie hotovej Linux-ovej distribúcie a samotná nie je generátorom distribúcií. Konfigurácia inštalácie pomocou Instalinux-u prebieha v niekoľkých krokoch:

- Výber distribúcie – je možné vybrať si z CentOS, Debian, Fedora, OpenSUSE, Scientific, Ubuntu, ich verzii a cieľovej hardvérovej architektúry (x86 alebo x86_64); v tomto kroku je tiež možné nastaviť hostname alebo IP adresu výsledného systému.
- Spôsob inštalácie – je závislý od predošlého kroku; na výber býva inštalácia z CD definovanej distribúcie, ktorou je však nutné fyzicky disponovať, alebo sieťová inštalácia cez http, či ftp server – podľa zvolenej distribúcie. V tomto kroku je tiež možné nastaviť aj preddefinovaný profil, teda nastavenia ďalších krokov, ktoré vytvorili a uložili iní používatelia a tiež sieťový adaptér pre sieťovú inštaláciu.
- Lokalizácia – nastavenie jazyka systému a klávesnice, časovej zóny a servera v závislosti od distribúcie, odkiaľ sa stiahnu údaje pre špecifikáciu distribúcie v ďalšom kroku.
- Špecifikácia – v tomto kroku je možné nastaviť balíčky. Výber je však obmedzený ponukou stiahnutou zo servera nastaveného v predchádzajúcom kroku a predstavuje množiny balíčkov nastaviteľné pri inštalácii z originálu danej distribúcie. Je tiež možné nastaviť spôsob rozdelenia disku – partícii a to aj aby rozdelenie a naformátovanie prebehlo automaticky pred inštaláciou systému.

- Záverečné nastavenia – dovoľujú nastaviť heslo pre root používateľa, nastaviť tiež jedného bežného používateľa a manuálne upraviť finálny inštalačný skript.

Výstupom tejto aplikácie je ISO obraz, ktorý buď nainštaluje systém automaticky pomocou siete a zvolených definícii počas procesu jeho nastavenia, alebo je potrebné k obrazu dodať inštalačné súbory zvolenej distribúcie, ktoré sa použijú ako zdroj.

Aplikácia je používateľsky ľahko ovládateľná, predstavuje dé facto grafické rozhranie, pre nástroje vyvinuté spoločnosťou HP – LinuxCOE. Umožňuje definíciu automatickej inštalácie, kde nie je po nabootovaní potrebný zásah používateľa, no neprináša žiadnu novú distribúciu.

4.1.2 Slax Build

Slax je live distribúcia Linux-u, spustiteľná priamo z bootovateľného média, bez inštalátora. Na hlavnej stránke tejto distribúcie je možné stiahnuť hotový ISO obraz, no tiež pohodlne vyskladať distribúciu založenú na jadre Slax-u. Samotná východzia distribúcia obsahuje jadro Slax-u, grafické prostredie, webový prehliadač, jednoduché kancelárske a multimediálne aplikácie a vývojárske nástroje. ISO obraz tejto distribúcie zaberá 200 MB. Maximálna redukcia – na jadro Slax-u zaberá 56 MB, čo predstavuje kernel skompilovaný so základnými nastaveniami (napríklad obmedzená podpora hardvéru) a výrazne redukovanú sadu aplikácii (napríklad absencia kompilačných nástrojov).

Slax dokáže byť teda veľmi kompaktný, no obmedzený z hľadiska možností nastavenia. Webové rozhranie ponúka len možnosť zvolenia balíčkov (pri tejto distribúcii sú nazývané modulmi) z vlastnej databázy, ktorá je značne obsiahla, avšak žiadnu konfiguráciu používateľov, lokalizácie a podobne. Keďže ide o live systém, je prirodzené, že sú vynechané možnosti nastavenia disku

a partícií.

4.1.3 SUSE Studio

Tento projekt ponúka vyspelé riešenie zostavovania a konfigurácie distribúcie. Po prihlásení sa do systému SUSE Studio má používateľ na výber možnosť zložiť si vlastnú distribúciu alebo vybrať zo širokej ponuky už vytvorených distribúcií. Tieto sú zväčsa vytvorené inými používateľmi a zvyčajne obsahujú stručný popis, hlavne charakteristiku využitia. V detailnom zobrazení distribúcie je možné zobraziť sadu nainštalovaných balíčkov a konfiguráciu systému, prehliadať zoznam starších verzií distribúcie, zobraziť diskusiu. Je tiež možné priamo kontaktovať autora alebo vytvoriť vlastnú novú distribúciu so základom zobrazenej. Jednotlivé distribúcie a balíčky je možné prehliadať – triediť aj na základe popularity.

Tvorba vlastnej distribúcie je používateľsky priateľská a prebieha v niekoľkých krokoch. V prvom kroku si používateľ vyberie cieľovú architektúru (x86 alebo x86_64), základ – šablónu pre svoju distribúciu, pričom na výber je:

- Just enough OS (JeOS) – distribúcia s minimom základných aplikácií
- Serverová distribúcia
- Minimálna distribúcia s GUI (IceWM, Gnome, KDE)

SUSE Studio ponúka možnosť doinštalovať ďalšie balíčky zo svojej databázy, tiež odinštalovať balíčky definované šablónou. Vyhodnocuje závislosti medzi balíčkami a v prípade zistenia chýbajúceho balíčka, ktorý je potrebný, upozorní používateľa. Pre distribúciu je ďalej možné nastaviť lokalizáciu a časové pásmo, sieť, používateľov (tiež ich heslá, domovský adresár a východziu shell aplikáciu) a priradenie do skupín používateľov, počiatočnú úroveň behu a východzieho používateľa, ktorý bude automaticky prihlásený do systému. V

online prostredí sa dá nastaviť aj databáza – v ponuke je mySQL a PostgreSQL, no je možné nastaviť iba používateľov a nahrať SQL skript, ktorý sa pri štarte systému vykoná. Tiež je možné definovať shell-ovské skripty, ktoré sa majú vykonať po zostavení alebo nabootovaní distribúcie. Do výslednej distribúcie má používateľ možnosť nahrať do ním určenej adresárovej štruktúry vlastné súbory alebo archívy.

Po nastavení distribúcie ju môže používateľ zostaviť do viacerých formátov. Momentálne sú podporované:

- live obraz pre USB zariadenia
- live ISO obraz
- obraz disku – komprimovaný programom gzip a inštalovateľný alebo bootovateľný ISO obraz – obe voľby však zmažú obsah na cieľovom disku a automaticky vytvoria a naformátujú partície bez možnosti nastavenia, pričom zaberú plnú kapacitu, čo nie je pre najmä skúsenejších používateľov často želané.
- obrazy pre cloud computing službu Amazon EC^2
- hotové obrazy pre virtuálne zariadenia ako VirtualBox, či VMWare

SUSE Studio je veľmi prepracovaná aplikácia. Ako základ však používa distribúciu openSUSE a nie je možné ako východziu nastaviť inú, no ponúka širokú škálu nastavení a je na používateľsky veľmi vysokej úrovni. Základný JeOS systém, podobne ako serverová distribúcia nevyužíva grafické rozhranie a live ISO obraz zaberá 155 MB, respektíve 173 MB pre serverovú distribúciu.

Výslednú vlastnú distribúciu je tiež možné online otestovať. Používateľ je však časovo obmedzený 15 minútami, čo postačuje len na úvodné zoznámenie sa s novým systémom a nie na kompletné otestovanie.

Kapitola 5

generatux

Generatux je výslednou aplikáciou tejto práce a je ďalším online nástrojom pre tvorbu a generovanie Linux-ových distribúcií. Pre použitie aplikácie je potrebná registrácia a následné prihlásenie. Po prihlásení dostane používateľ prístup ku svojim projektom, ktoré môže pridávať alebo upravovať. Projektu je možné zadať základné informácie ako meno, verziu, typ, popis a poznámku projektu. Typ projektu môže byť súkromný alebo verejný, čo značí obmedzenie prístupu k projektu ostatnými používateľmi, pričom súkromné projekty nie sú pre ostatných používateľov vôbec viditeľné. Verejné projekty sú viditeľné a síce nie je možné ich ostatnými konfigurovať, no ak je ISO obraz projektu vygenerovaný, je možné ho stiahnuť.

5.1 Administrátorské prostredie

Používateľom je pre aplikáciu generatux možné pridať administrátorské práva. V praxi to znamená, že administrátor – používateľ s týmito právami môže vytvárať preddefinície, databázu balíčkov, spravovať dostupné súborové systémy a body pripojenia. Pre súborové systémy je okrem názvu možné definovať popis, ktorý sa zobrazí používateľovi pri konfigurácii partícií a príkaz, ktorým sa daný súborový systém na partícii počas inštalácie vytvorí. Nástroj,

ktorý daný súborový systém vytvorí však musí byť obsiahnutý v počiatočnom RAM disku inštalácie.

Bodom pripojenia je takisto možné okrem názvu, ktorý reprezentuje samotný bod, definovať popis.

Balíček má atribúty – meno, verzia, popis, príznak základného balíčka, typ a súbor. Balíčky, ktoré obsahuje JeOS a teda aj všetky ostatné distribúcie sú označené ako základné. Typ balíčka môže byť systémový, utility, sieťový, bezpečnostný, grafický alebo vývojársky. Ako súbor balíčka je potrebné nahrať už zostavený, skompilovaný balíček vo forme skomprimovaného archívu vo formáte gzip; to však nie je nutné pre základné balíčky, pretože tieto už sú obsiahnuté v JeOS systéme. Tento súbor sa potom použije pri zostavovaní ISO obrazu a samotnej inštalácii.

Hlavnú časť administrátorského prostredia tvoria preddefinície – predpripravené distribúcie. Základ pre každú preddefiníciu tvorí vopred zostavený JeOS systém a nové preddefinície sú jeho rozšírením. Preddefiníciám je možné v základných úpravách zvoliť názov, ikonku, popis a internú poznámku (táto sa nebude pri zostavovaní systému používateľom zobrazovať). Dôležitý je však výber balíčkov, kde sa zobrazí zoznam dostupných balíčkov, ktorý je možné triediť podľa ich typu. Tieto balíčky je potom možné priradiť danej preddefinícii. V zozname sa však zobrazujú iba balíčky, ktoré nie sú označené ako základné, pretože tie sú v konfigurovanej preddefinícii štandardne zahrnuté.

5.2 Konfigurácia projektu

Konfigurácia prebieha v dvoch úrovniach. V prvej úrovni, pri novom projekte, je potrebné zvoliť predpripravenú distribúciu – preddefiníciu. V základe je na výber JeOS „Just Enough Operating System“ systém, pričom v administrátorskom prostredí je, ako bolo spomenuté, možné definovať ďalšie.

5.2.1 JeOS systém

JeOS systém, ktorý používa generatux ako základ, je systém vybudovaný podľa Linux From Scratch [Bee10] s niektorými dodatočnými aplikáciami, napríklad pre shell. Tento systém tvorí tiež východziu distribúciu pre ďalšie preddefinície, ktoré ho rozširujú zvolenými balíčkami. JeOS ako taký nemá presnú definíciu balíčkov, ktoré má obsahovať. Má to však byť jednoduchý malý systém pre nejaké, zvyčajne autorom určené využitie. Generatux JeOS je cielený ako základ pre ďalšie rozširovanie, no už bez ďalších zmien je vhodný pre vývoj programov, úpravu textov a obsahuje jednoduché sieťové aplikácie.

Balíčky, ktoré tento systém obsahuje sú:

Systémové

- Linux Kernel – Samotný kernel operačného systému, opísaný v časti 2.1
- Module-Init-Tools – Sada nástrojov pre správu modulov kernelu. Obsahuje programy pre nahratie, odobratie kernel modulu, zobrazenie nahratých modulov, informácií o nich a závislostí medzi nimi.
- Sysvinit – Balíček obsahujúci základné programy pre chod systému; nachádza sa medzi nimi napríklad program `init`, spomínaný v časti o štarte systému (2.4.3).
- Procps – Sada nástrojov pre monitorovanie procesov, manipuláciu s nimi a zisťovanie stavu systému (napríklad aktuálne zaťaženie systémových prostriedkov).
- Psmisc – Sada nástrojov pre zisťovanie informácií o bežiacich procesoch.
- Sysklogd – Obsahuje programy pre zaznamenávanie systémových správ.

- Udev – Balíček obsahuje obslužné programy pre statické a dynamické pripájanie zariadení.
- Util-linux – Sada systémových nástrojov rôzneho využitia, hlavne však pre správu partícií a vytváranie a pripájanie súborových systémov.
- Glibc – Hlavná knižnica programovacieho jazyka C. Obsahuje funkcie pre alokovanie pamäte, prácu so súbormi a adresármi, prácu s reťazcami a regulárnymi výrazmi a iné fundamentálne funkcie.
- E2fsprogs – Programy pre správu ext2, ext3 a ext4 súborových systémov spomínaných v časti 2.2.1.
- ReiserFS – Programy pre správu ReiserFS súborového systému z časti 2.2.1.
- XFS – Programy pre správu XFS súborového systému z časti 2.2.1.
- GRUB – Boot loader, spomínaný v časti 2.4.2.
- Bash – Bourne-Again Shell, spomínaný v časti 2.3.
- Coreutils – Obsahuje programy pre nastavovanie, zobrazovanie a správu súborov (zobrazenie aktuálneho adresára, zobrazenie obsahu adresára, vymazanie súboru, či adresára, nastavavanie práv a iné) ale aj ďalšie užitočné nástroje.
- GDBM – GNU Database Manager – jednoduchá, ale veľmi rýchla databáza na princípe kľúč - hodnota. Využíva ju balíček Man-DB.
- Man-DB – Obsahuje programy pre vyhľadanie a zobrazenie manuálov.
- Man-pages – Základný manuál Linux-u.
- Kbd – Podporné programy pre klávesnicu, množina fontov.

- Readline – Obsahuje knižnice pre zjednodušenie práce s príkazovým riadkom (história príkazov a podobne).
- Shadow – Obsahuje programy pre bezpečnú správu hesiel.
- Csh – C Shell, spomínaný v časti 2.3.
- Tcsh – Tenex C Shell, spomínaný v časti 2.3.
- Ksh – Korn Shell, spomínaný v časti 2.3.

Utility

- Bzip2 – Obsahuje komprimačné a dekomprimačné nástroje pre formát bz2.
- Diffutils – Nástroje pre porovnávanie súborov a adresárov, je vhodný napríklad pre vytváranie záplat programov, či balíčkov.
- Findutils – Balíček obsahuje programy pre vyhľadávanie súborov.
- File – Program pre zisťovanie typov súborov.
- Gettext – Obsahuje programy pre podporu lokalizácie viacerých balíčkov.
- Grep – Množina nástrojov pre vyhľadávanie v súboroch a textoch.
- Groff – Obsahuje programy pre spracovanie a formátovanie textu.
- Gzip – - Obsahuje komprimačné a dekomprimačné nástroje pre formát gz.
- Less – Program pre zobrazovanie textových súborov.
- Ncurses – Obsahuje knižnice poskytujúce rozhranie pre textové terminálové aplikácie; je používaný viacerými balíčkami.

- Patch – Program pre aplikáciu „patch“ súborov.
- Pkg-config – Nástroj pre zisťovanie dependencií a ciest k potrebným knižniciam pri inštalácií balíčkov.
- Sed – Nástroj pre spracovanie textových súborov.
- Tar – Archivačný program.
- Texinfo – Sada programov pre zobrazenie a úpravy informačných súborov - stránok; je často využívaný inštalačnými procedúrami balíčkov.
- Vim – Textový editor.
- XZ Utils – Obsahuje komprimačné a dekomprimačné programy pre formáty xz a lzma.
- Zlib – Komprimačné a dekomprimačné funkcie.

Vývojárske

- Autoconf, Automake, Libtool, Binutils, Bison, GCC, m4 – Nástroje pre kompiláciu spomenuté v časti 3.2.
- Flex – Obsahuje program pre tvorbu programov rozoznávajúcich výrazy v texte.
- Gawk – Balíček s nástrojmi pre vytváranie skriptov pracujúcich s textovými súbormi.
- Make – Program pre spúšťanie `makefile` skriptov.
- GMP – Obsahuje matematické knižnice. Je vyžadovaný GCC.
- MPC – Obsahuje knižnice pre aritmetiku komplexných čísel; vyžadovaný balíčkom GCC.

- MPFR – Ďalšie matematické knižnice vyžadované GCC.
- Perl – Nástroje pre programovací jazyk PERL; vyžadovaný pre inštaláciu niektorých balíčkov.

Sieťové

- Iana-etc – Balíček konfiguračných súborov sieťových služieb a protokolov.
- Inetutils – Obsahuje základné sieťové programy.
- IProute2 – Množina sieťových programov pre IPv4 a IPv6 protokol.

Po zvolení príslušného základu je možné konfigurovať ďalšie časti projektu:

- Partície
- Používatelia
- Lokalizácia
- Východzia úroveň behu
- Prispôsobenie

Ak ešte používateľ nezadal voľnú kapacitu a typ disku (/dev/sda pre SCSI disk alebo /dev/hda pre IDE disk), je nútený tieto údaje nastaviť. V ďalšom kroku potom môže nastaviť partície na tomto disku a to zadaním menovky a typu (primárna alebo rozšírená) partície, bodu pripojenia (mount point), typu súborového systému a veľkosti partície. Pri tvorbe partícií sa vykonáva kontrola, či používateľ napríklad nevytvoril dve swap partície, využitie bodu pripojenia najviac jednou partíciou a podobne.

Východzie nastavenie každého projektu obsahuje používateľa `root`. Tento používateľ je chránený – nezmazateľný a je možné mu nastaviť iba heslo a shell aplikáciu. Je však tiež potrebné pridať aj „bežného“ používateľa, pretože z bezpečnostného hľadiska nie je možné, aby root bol jediným používateľom definovaným v systéme. Systém vykonáva kontrolu unikátnosti mena používateľa a tvaru povinných atribútov (regulárny výraz pre tvar mena používateľa je možné nastaviť v kóde; štandardne je nastavený na reťazec alfanumerických znakov s pomlčkou začínajúci písmenom).

V rámci konfigurácie lokalizácie je možné nastaviť jazyk klávesnice a časovú zónu.

Východzia úroveň behu predstavuje úroveň, do ktorej bude systém po nainštalovaní a nabootovaní prepnutý. Úrovniam behu sa venuje časť 2.4 o bootovaní a štarte systému.

V rámci dodatočného prispôsobenia distribúcie je možné ju čiastočne priestorovo optimalizovať vymazaním manuálov a s nimi príbuzných aplikácií. Tiež ak používateľ nechce používať sieťové alebo vývojové aplikácie, tieto nebudú do výsledného systému zahrnuté.

5.3 Generovanie projektu

Ak používateľ prešiel všetkými krokmi konfiguračného procesu, je možné vygenerovať inštalačný ISO obraz distribúcie; v opačnom prípade na to bude pri pokuse o vygenerovanie projektu upozornený a zobrazí sa mu zoznam nenastavených atribútov. Samotné generovanie prebieha na pozadí a nie je potrebná žiadna ďalšia intervencia zo strany používateľa. Proces generovania prebieha v niekoľkých krokoch vykonávaných kódom aplikácie a programami hostiteľského systému, na ktorom aplikácia beží.

5.3.1 Príprava RAM disku

Generatux ako boot loader ISO obrazu používa ISOLINUX. Táto jednoduchá aplikácia dokáže v úvode zobraziť textovú správu s informáciami, v ktorých môže byť napríklad ponúknutý zoznam Linux-ových systémov pre nabootovanie. Každá možnosť potrebuje mať nadefinovaný kernel a RAM disk, s ktoými sa príslušný systém nabootuje. ISOLINUX je však možné nakonfigurovať aj tak, aby automaticky spustil nadefinovaný východzí systém, čo je využité aj pri obrazoch tvorených generatux-om.

Generatux používa jednoduchý RAM disk na základe busybox-u – balíčka, ktorý enkapsuluje základné utility najmä pre prácu so súbormi a diskom a shell. Bootovanie z inštalačného média vytvoreného generatux-om prebieha podobne ako bootovanie z pevného disku, avšak v závere je zavolaný skript `init`, ktorý následne po inicializácii systému, zavolá skript `install`. Tento skript je pri vytváraní obrazu generovaný dynamicky na základe konfigurácie zvolenej používateľom. Obsahuje funkcie pre vytvorenie a naformátovanie partícií s definovanými atribútmi vo webovom rozhraní – veľkosťou, súborovým systémom a podobne, a tiež funkcie pre samotnú inštaláciu systému.

RAM disk, ako bolo spomenuté, je vlastne počiatočnou adresárovou štruktúrou. Generatux teda v procese generovania skopíruje generický predpripravený RAM disk do projektového adresára (adresár, kde sa generujú súbory pre výsledný obraz), vygeneruje skript `install`, a keďže je potrebné aby bol RAM disk reprezentovaný jedným súborom, vytvorí sa jeho komprimovaný archív.

Kernel, ktorý daný RAM disk použije pri bootovaní, v tejto fáze nie je taký dôležitý a nebudeme sa ním príliš zaoberať. Je to však iný kernel aký bude používať výsledný inštalovaný systém a stačí, ak to bude napríklad kernel prevzatý z hostiteľského systému.

5.3.2 Príprava distribúcie

Do projektového adresára sa najprv nakopíruje pripravená adresárová štruktúra JeOS distribúcie. Následne sú v tejto skopírovanej distribúcii upravené základné systémové konfiguračné súbory:

- fstab – na základe zvolených partícií sa do tohto súboru dogenerujú príslušné riadky pre pripojenie systémov súborov; formát súboru: [Wike]
- passwd – podľa vytvorených používateľov sa upraví tento súbor obsahujúci ich zoznam; formát súboru: [Wikg]
- group – podľa vytvorených používateľov a im priradených skupín sa upraví tento súbor obsahujúci ich zoznam; formát súboru: [Wikf]
- shadow – obsahuje heslá používateľov a ich platnosť; formát súboru: [Wikh]
- timezone – časová zóna sa nastaví podľa preferencií používateľa

Po týchto úpravách sa adresárová štruktúra distribúcie skomprimuje do jedného výsledného súboru.

Ostatné balíčky podľa zvolenej preddefinície systému, ktoré netvoria JeOS systém sa nachádzajú predkompilované a skomprimované vo vyhradenom adresári v rámci web aplikácie, odkiaľ sú skopírované do podadresára v projektovom adresári.

5.3.3 Vyčistenie projektového adresára a vygenerovanie ISO obrazu

V tomto poslednom kroku generovania sa najprv odstránia nepotrebné adresáre a súbory z projektového adresára. Po predošlých dvoch krokoch totiž v projektovom adresári zostali adresárové štruktúry distribúcie a RAM disku,

ktoré nie sú viac potrebné a zaberali by miesto vo výslednom obraze. Po ich odstránení sa príkazom `mkisofs` s príslušnými parametrami vytvorí ISO obraz obsahu projektového adresára, ktorý si môže tvorca projektu, ale aj ktokoľvek iný, ak je projekt verejný, po jeho kompletnom vytvorení stiahnuť v prostredí generatux-u.

5.4 Inštalácia systému

Vygenerovaný a stiahnutý ISO obraz je ďalej možné napáliť na CD, či DVD nosič, no pohodlnejšia možnosť je vyskúšať ho vo virtualizovanom prostedí, aké poskytujú napríklad aplikácie VMWare alebo VirtualBox. Posledne menovaná aplikácia tiež slúžila v práci ako samotné vývojové, ale aj testovacie prostedie.

Po spustení hardvéru, s pripojeným vygenerovaným ISO obrazom alebo médiom v mechanike, sa spustí boot loader ISOLINUX, ktorý sme nakonfigurovali aby bez zásahu používateľa spustil vygenerovaný RAM disk s počiatočným kernelom. Po spustení tohto úvodného systému sa zobrazí dialóg, kde používateľ môže zvoliť automatickú inštaláciu, spustiť shell alebo rebootovať systém. Posledné dve možnosti sú zrejmé, zaujímavá je prvá možnosť.

Automatická inštalácia prebehne v troch krokoch. V prvom kroku dôjde k nastaveniu partícii disku a ich naformátovaniu podľa zvolených súborových systémov. Tieto systémy sa po vytvorení pripoja na definované body pripojenia aby boli dostupné. V druhom kroku už dôjde k samotnej inštalácii distribúcie. Do súborového systému pripojeného na koreňový adresár sa skopíruje dekomprimovaný archív a následne balíčky, ktoré nie sú v základnom JeOS systéme. Ako finálny krok sa nakoniec nastaví boot loader GRUB, uvoľní sa inštalačné médium a počítač sa rebootuje do nového nainštalovaného systému.

Kapitola 6

Záver

V súčasnosti existuje veľa Linux-ových distribúcií, vytvorených za rôznymi účelmi. Dôvodom zvyčajne býva využitie danej distribúcie v praxi. Modulárna štruktúra Linux-u, ktorú sme v práci popísali, umožňuje vytvoriť operačný systém na jeho báze ako pre jednoúčelové, tak aj pre vysokokomplexné zariadenia.

Spočiatku sa takéto distribúcie tvorili ručne, bez automatizovaných procesov. Postupom času však vznikali aplikácie, ktoré umožňovali ich tvorbu zjednodušiť a v súčasnosti, vďaka rozmáhajúcemu sa internetu, sa skladanie distribúcií stalo triviálnym postupom pre používateľa, ktorý si dokáže vlastnú distribúciu vytvoriť sám pomocou webového rozhrania.

V úvodných častiach práce sme si ukázali ako Linux funguje, z akých komponentov sa skladá a čo je potrebné pre vývoj Linux-ovej distribúcie. Tieto poznatky sme využili pri tvorbe online aplikácie, ktorá dokáže postaviť takúto distribúciu na základe používateľom definovaných atribútov.

Počas tvorby práce sme však menili prvotné ciele:

- Na základe získaných poznatkov sme zistili, že nie je potrebné nechať používateľovi voľbu boot loaderu. GRUB je plne vyhovujúci a poskytuje viac možností ako zastaralý LILO boot loader.

- Podobne, možnosť výberu kernelu je zbytočná. Cena za skomplikovanie tvorby aplikácie s touto možnosťou je väčšia ako jej prínos.

- Balíčkovací systém takisto stráca pri generovaných distribúciách svoj význam. Distribúcia vygenerovaná pomocou takéhoto nástroja by totiž mala byť pre používateľa finálna, bez potreby niečo meniť.

Pri zhodnocovaní existujúcich aplikácií sme takisto dospeli k záveru, že v tejto fáze aplikácie nie je podstatné vytvoriť rozsiahlu databázu balíčkov. Online nástroje obsahujúce stovky balíčkov, z ktorých sa dá poskladať výsledný systém totiž už existujú. Zamerali sme sa tak na nový cieľ – vytvorenie plne automatickej inštalácie, čo sa nám podarilo. Administrátorské prostredie však dovoľuje pridávať nové balíčky a priraďovať ich jednotlivým preddefinovaným distribúciám, ktoré si môže používateľ vybrať.

Ďalšou devízou aplikácie generatux je otvorenosť zdrojového kódu. Môže teda slúžiť ako zdroj pre podobný systém alebo poskytuje priestor pre jej samotné rozšírenie.

6.1 Rozšírenia

Jedným z možných rozšírení je vytvoriť databázu balíčkov. Tento proces je však časovo veľmi náročný a vyžaduje taktiež implementovať systém vyhodnocovania závislostí balíčkov. Táto funkcionalita je síce zahrnutá v cieľoch práce, no aplikácie, ktoré ju podporujú, sú známe a tak sa tento cieľ stal minoritným. Ak by však takáto databáza existovala, bolo by možné vytvoriť širokú kolekciu predpripravených distribúcií.

Ďalším možným rozšírením je v oblasti konfigurácie distribúcie. Šlo by o nastavovanie špeciálnych aplikácií ako je HTTP server, FTP server, mySQL server, či aplikáciu `samba` pre zdieľanie adresárov. Bolo by ale potrebné vybrať exaktnú množinu týchto aplikácií a zistiť možnosti konfigurácie.

Literatúra

[Bee10] Gerard Beekmans. *Linux From Scratch 6.8*. Iuniverse Inc 2000, 2010.

[Bri] Tony Brijeski. *Remastersys*. `http://remastersys.sourceforge.net/remastersystool.html`.

[Chi04] Brandon Chisham. Introduction to linux, 2004.

[DPB05] Marco Cesati Daniel P. Bovet. *Understanding the Linux Kernel, 3rd Edition*. O'Reilly, 2005.

[Fle05] Michael Flenov. *Hacker Linux Uncovered*. A-LIST Publishing, 2005.

[Fou] The Linux Foundation. *Filesystem Hierarchy Standard*. `http://www.pathname.com/fhs/`.

[Gar08] Machtelt Garrels. Introduction to linux, 2008.

[GNU] GNU. *GNU Software*. `http://www.gnu.org/software/`.

[GOL] *General overview of the Linux file system*. `http://tldp.org/LDP/intro-linux/html/sect_03_01.html`.

[Hin] Martin Hinner. *Filesystems List*. `http://tldp.org/HOWTO/Filesystems-HOWTO.html`.

[Lam] Chris Lamb. *Live Magic*. `http://chris-lamb.co.uk/projects/live-magic/`.

[Lov05] Robert Love. *Linux Kernel Development Second Edition*. Sams Publishing, 2005.

[MKD05] Matt Welsh Matthias Kalle Dalheimer. *Running Linux, 5th Edition*. O'Reilly, 2005.

[Neg06] Christopher Negus. *Linux Bible 2006 Edition*. Wiley Publishing, Inc., 2006.

[oL] First Dutch International Symposium on Linux. *Design and Implementation of the Second Extended Filesystem*. `http://e2fsprogs.sourceforge.net/ext2intro.html`.

[Proa] Binh Nguyen The Linux Documentation Project. *Linux Filesystem Hierarchy*. `http://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/Linux-Filesystem-Hie`

[Prob] The Fedora Unity Project. *Revisor*. `http://revisor.fedoraunity.org/`.

[Proc] The Linux Documentation Project. *Introduction to Linux Loadable Kernel Modules*. `http://tldp.org/HOWTO/Module-HOWTO/x73.html`.

[Ray03] Steven Raymond. *The Art of Unix Programming*. Addison Wesley, 2003.

[Rus98] David A. Rusling. Linux kernel, 1998.

[Sch04] Carla Schroder. *Linux Cookbook*. O'Reilly, 2004.

[SG03] Gene Spafford Simson Garfinkel, Alan Schwartz. *Practical Unix & Internet Security, 3rd Edition*. O'Reilly, 2003.

[SGI] SGI. *XFS: A high-performance journaling filesystem*. `http://oss.sgi.com/projects/xfs/`.

[Sob05] Mark G. Sobell. *A Practical Guide to Linux® Commands, Editors, and Shell Programming*. Prentice Hall PTR, 2005.

[Tan] Andrew Tanenbaum. *Linux is obsolete*. `http://oreilly.com/catalog/opensources/book/appa.html`.

[Tho05] Keir Thomas. *Beginning SUSE Linux: From Novice to Professional*. Apress, 2005.

[War04] Brian Ward. *How Linux Works: What Every Super-User Should Know*. No Starch Press, 2004.

[Wika] Wikipedia. *Autoconf*. `http://en.wikipedia.org/wiki/Autoconf`.

[Wikb] Wikipedia. *Bourne shell*. `http://en.wikipedia.org/wiki/Bourne_shell`.

[Wikc] Wikipedia. *Btrfs*. `http://en.wikipedia.org/wiki/Btrfs`.

[Wikd] Wikipedia. *C shell*. `http://en.wikipedia.org/wiki/C_shell`.

[Wike] Wikipedia. */etc/fstab*. `http://en.wikipedia.org/wiki/Fstab`.

[Wikf] Wikipedia. */etc/group*. `http://en.wikipedia.org/wiki//etc/group`.

[Wikg] Wikipedia. */etc/passwd*.
`http://en.wikipedia.org/wiki/Passwd_(file)`.

[Wikh] Wikipedia. */etc/shadow*.
`http://en.wikipedia.org/wiki/Shadow_password`.

[Wiki] Wikipedia. *ext4*.
`http://en.wikipedia.org/wiki/Ext4`.

[Wikj] Wikipedia. *GNU Build System*.
`http://en.wikipedia.org/wiki/GNU_build_system`.

[Wikk] Wikipedia. *Korn shell*.
`http://en.wikipedia.org/wiki/Korn_shell`.

[Wikl] Wikipedia. *Linux kernel*.
`http://en.wikipedia.org/wiki/Linux_kernel`.

[Wikm] Wikipedia. *Reiser4*.
`http://en.wikipedia.org/wiki/Reiser4`.

[Wikn] Wikipedia. *Tanenbaum–Torvalds debate*.
`http://en.wikipedia.org/wiki/Tanenbaum-Torvalds_debate`.

[Wiko] Wikipedia. *tcsh*.
`http://en.wikipedia.org/wiki/TENEX_C_shell`.