Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Praktické úlohy konfigurace a správy GNU/Linux Ubuntu

Bakalářská práce

Autor: Michaela Kratochvílová
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. Mgr. Josef Horálek, Ph.D.
Odborný konzultant: Ing. Jiří Bönsch

Hradec Králové Duben 2024

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně a s použitím uvedené literatury.

V Hradci Králové dne 25.4.2024

Michaela Kratochvílová

Poděkování:

Děkuji vedoucímu bakalářské práce doc. Mgr. Josef Horálek, Ph.D. za metodické vedení práce a trpělivost. Děkuji také Ing. Jiřímu Bönschovi za podnětné rady a pomoc s touto prací. Dále také děkuji mé rodině, která mi pomáhala a podporovala ve vypracování práce.

Abstrakt

Tato bakalářská práce představuje základy práce s distribucí operačního systému Linux zvanou Ubuntu. Čtenářům vysvětluje, jak nakonfigurovat a spravovat systém. Laboratorní úlohy obsahují kromě teoretického výkladu také praktické úkoly a otázky na ověření znalostí získaných z laboratorních úloh. Teoretický výklad důkladně popisuje postupy, které čtenář může využít při používání Ubuntu a také vysvětluje různé možnosti ke kterým by mohly být dané příkazy a postupy využity. Nejčastější postupy jsou důkladněji vysvětleny a měly by být při čtení prováděny, aby si je čtenář lépe osvojil. Praktické úlohy jsou podobné těm uvedeným v teoretickém výkladu a čtenář by si tím měl ověřit, zda chápe teoretický výklad a dokáže ho také provést. Teoretické otázky související s danou laboratorní úlohou by měl být čtenář schopen zodpovědět po přečtení dané laboratorní úlohy. Cílem je, aby čtenář po přečtení dokázal provést základní postupy při práci s touto distribucí a také chápal teoretický výklad, jenž by následně dokázal předvést.

Klíčová slova:

Linux, distribuce Ubuntu, administrace systému, konfigurace systému Ubuntu

Abstract

Title: Practical tasks about configuration and administration of GNU/Linux Ubuntu

This Bachelors thesis presents the basics of working with the distribution of the Linux operating system called Ubuntu. It explains to readers how to configure and manage said system. In addition to the theoretical explanation, the laboratory tasks also contain practical tasks and questions to verify the knowledge obtained from the laboratory tasks. The theoretical explanation thoroughly describes the procedures that the reader can use when using Ubuntu and explains the various possibilities for which the given commands and procedures could be used. The most common procedures are explained more thoroughly and should be performed while reading so that the reader becomes familiar with them. The practical tasks are alike to those given in the theoretical explanation. The reader should verify that he understands the theoretical explanation and can execute said tasks by performing the practical tasks. The reader should be able to answer the theoretical questions related to the given laboratory task after reading the laboratory task. The goal is for the reader, after reading, to be able to execute the basic procedures for working with this distribution and also to understand the theoretical interpretation, which he would then be able to demonstrate.

Key words:
Linux, distribution Ubuntu, system administration, Ubuntu configuration

Obsah

Seznam obrázků

1 Úvod

Uživatele počítačů v dnešní době můžeme rozdělit podle jejich používaného operačního systému a to macOS, Windows a Linux. Někteří uživatelé využívají i vícero operačních systémů zároveň.

Tato práce je zaměřená na uživatele systému Linux a jeho distribucí. Tato skupina uživatelů není nejpočetnější, ale je nejvíce zaměřená na ochranu svých dat a větší možnosti úprav svého operačního systému.

Operační systém Linux, který vznikl v roce 1991, je založený na principech systému Unix a jedná se o jeden z nejrozšířenějších a nejvýznamnějších open-source systémů. Linux je také rozdělen do distribucí, z nichž mezi nejznámější patří Ubuntu, Fedora nebo Debian. Linux je využíván na počítačích a serverech, ale také na mobilních zařízeních využívajících Android.

V této bakalářské práci jsou vysvětleny základní postupy pro práci se systémem Linux, představené na distribuci Ubuntu a formátované do laboratorních úloh. Tyto postupy jsou vytvořeny především pro začátečníky, a proto nemusí být pro pokročilé uživatele přínosné. Laboratorní úlohy také obsahují praktické a teoretické otázky na konci, které mají ověřit znalosti čtenáře obsažené v dané úloze. Po přečtení a vypracování laboratorních úloh by čtenáři měli být schopni práce s distribucí Ubuntu na svém systému.

Postupy popsané v této práci v době psaní bakalářské práce jsou funkční a využívané, v době vzniku práce ale nemusí tomu tak být v době čtení této práce. Dále jsou některé odborné výrazy použité v angličtině, jelikož se tak běžně využívají a je vhodné se s nimi seznámit v jejich původní podobě. V této práci jsou využity ve větší části Cisco kurzy pro získání obecných informací. [1–3]

2 Cíl a metodika práce

Běžný uživatel počítače nemá potřebu vybírat si mezi různými operačními systémy, uživatelé operačního systému Linux tedy většinou spadají do kategorie odborníků v oboru výpočetní techniky. Využití Linuxu by přitom mohlo být pro běžné uživatele přínosem. Operační systém si uživatel může přizpůsobit svým požadavkům. Systém je stabilní, bezpečný a většinou úplně zdarma. Linux je také známý jako velmi výkonný i na starších zařízeních, jelikož nezabírá mnoho paměti a je zaměřen na optimalizaci výkonu.

Cílem této bakalářské práce je představit začátečníkům práci s Ubuntu formátovanou do laboratorních úloh zabývajících se různými tématy. Dané laboratorní úlohy nejdříve vysvětlují jejich téma na praktické ukázce doplněné o teorii a další možnosti příkazů. Na konci každé laboratorní úlohy jsou otázky na ověření získaných znalostí a také praktická úloha, kterou by čtenář měl být schopen po přečtení laboratorní úlohy samostatně provést. K praktické úloze je přidáno i její řešení, pokud by si s tím čtenář neměl rady.

3 Linux

Linux je open-source operační systém, vytvořený v roce 1991 Linusem Torvaldsem a roku 1992 licencovaný pod GNU General Public License. Tato licence umožňuje uživatelům využívat, upravovat a redistribuovat zdrojový kód systému. Linux je jeden z nejvíce stabilních, bezpečných a spolehlivých operačních systémů, jenž je hojně využíván na serverech, superpočítačích a v podnikových prostředích. Kromě počítačů využívajících systém Linux také mezi jeho uživatele řadíme ty, kteří používají Android zařízení, jelikož základem operačního systému Android je právě Linux. [4]

Pro více funkcí je možné si nainstalovat rozšiřující distribuce, které již systém Linux obsahují. Distribuce jsou kolekce balíčků a aplikací na standardním Linuxu, cílené na různé potřeby uživatelů. Liší se poskytovanými službami, procesem instalace a aktualizace aplikací, poskytnutím podpory a aktualizací systému. Mezi populární distribuce patří například Red Hat Enterprise Linux, CentOS, Fedora, SUSE, Debian, Ubuntu a Arch. [5]

3.1 *Historie Linuxu*

Začátky Linuxu se se datují ještě před rok 1991. Počátečním bodem je vznik operačního systému Unix v 60. a 70. letech 20. století. Unix vyvinuli Ken Thompson a Dennis Ritchie. Jedná se o operační systém s hierarchickým souborovým systémem, umožňující spuštění více souběžných úloh, tzv. multitasking, a přístup více uživatelů na jednom zařízení. Linux z Unixu vychází, dokonce i jméno je spojením křestního jména Linuse Torvaldse a systému Unix.

Dalším bodem je operační systém Minix, který navrhl v 80. letech profesor Andrew S. Tanenbaum. Minix je založen na Unixu a jeho cílovou skupinou původně měla být akademická sféra, později byl rozšířen i pro uživatele mimo tuto sféru, avšak nyní již není dále vyvíjen.

V roce 1991 Torvalds zveřejnil Linux. Zpočátku měl být bezplatnou alternativou k operačnímu systému Minix. Později v roce 1993 byla vydána první komerční distribuce Linuxu zvaná Slackware. [6]

Dále se Linux rozrůstal díky dalším již zmíněným distribucím a také zásluhou oddané komunity vývojářů.

V roce 2005 Google Inc. koupil společnost Android Inc., která se následně rozhodla založit svůj operační systém na bázi Linuxu. V roce 2012 se Android stal nejpopulárnějším operačním systémem na mobilních zařízeních a dodnes tuto pozici drží. [7]

Mezi další zařízení, na kterých můžeme najít Linux, patří například herní konzole Steam Deck, jež používá distribuci SteamOS, vyvíjenou společností Valve. [8] Dále je Linux používán i ve Smart TV, jelikož některé z nic mají nainstalovaný operační systém Android TV nebo Tizen OS. [9]

3.2 *Systém verzování*

V době, kdy Linux upravoval pouze sám Torvalds, byl verzovací systém složen z rostoucích proměnných od nuly, což pro tehdejší potřeby stačilo. Později se však k němu připojilo více vývojářů, tudíž bylo třeba systém pro přehlednost změnit. Od Linuxu verze 1.0 se začal používat systém se třemi proměnnými a syntaxí *A.B.C*. Proměnná A reprezentuje hlavní verzi, proměnná B vedlejší a proměnná C reprezentuje číslo revize. Jako příklad bude uvedena kernel verze 1.1.95, jež je v prvním hlavním vydání, v prvním vedlejším vydání a v 95. revizi. [10]

Od verze 2.4 byla přidána čtvrtá proměnná, která reprezentuje vydání patche[1] a tento systém byl používán až do verze 2.6. Linux verze 3.0 začal používat nový systém verzování. Čtvrtá proměnná již není používána a verzování se vrátilo zpět k syntaxi *A.B.C*. Dále se také začaly používat pre-patch verze kernelu pro vývoj, označeny zkratkou *rc*. Číslo hlavní verze se zvýší, pokud počet vedlejších verzí dosáhne 20. Toto pravidlo je také používáno pro číslo kernel verze. V době psaní této práce je hlavní verze kernelu na čísle 5. [10]

[1] Patch je aktualizace softwaru, která se většinou provádí z důvodu bezpečnosti, opravy chyb nebo kvůli přechodu na novější verzi softwaru. [10]

3.3 *Patch systém*

Patchování je proces úpravy kódu pro opravy chyb nebo pro upgrade softwaru. Tyto úpravy jsou důležité pro bezpečnost zařízení, jelikož vynalézaví útočníci stále vyvíjí nové útoky na zranitelná místa softwaru. [11]

Patch je textový dokument, který obsahuje rozdíly mezi dvěma verzemi zdrojového kódu, z nichž první verze obsahuje původní kód a druhá jeho upravenou podobu. V Linuxu se vytváří pomocí **diff** programu. Pro vytvoření správného patche je třeba porovnávat patřičný zdrojový soubor s jeho novou verzí. Pro aplikování patche se použije **patch** program. Tento program si přečte **diff** soubor a aplikuje změny do zdrojového souboru. Pro vrácení již provedených změn je používán argument **-R** programu **patch**. [12]

3.4 *Ubuntu*

Ubuntu je jedna z distribucí Linuxu vytvořená v roce 2004 skupinou Canonical Ltd. Tento operační systém jako první zavedl pravidelné zveřejnění vydání, a to každých 6 měsíců. Dále každé čtvrté vydání, tj. jednou za dva roky, je dlouhodobě podporováno, a to až na 10 let. Tato vydání jsou označena jako LTS – long term support. [13]

3.4.1 Základní architektura

Jádro Linuxu má monolitickou architekturu, což znamená, že v režimu jádra běží všechny úlohy v jednom programu, tudíž není potřeba využívat mechanismy vnitro-procesové komunikace. Monolitická jádra také díky tomuto nabízí vyšší výkon a rychlejší provádění systémových volání. Jsou dále jednodušší na údržbu a implementaci. Ovšem mají i své slabé stránky, například zvýšené bezpečnostní riziko. Jelikož program běží v režimu jádra, může se případný útočník dostat do jádra a zničit celý systém či ohrozit integritu dat. Další nevýhodou je, že chyba jedné služby může napáchat škody v systému. [14]

3.4.2 Vnější rozhraní jádra

Virtuální paměť operačního systému je rozdělena do dvou režimů. Prvním z nich je režim jádra, který je využíván na spouštění ovladačů, jádra operačního systému

a dalších rozšíření jádra. Tento paměťový prostor je vyhrazen na vykonávání a poskytování služeb jádra operačního systému. Druhým režimem je uživatelský režim. Tento režim je využíván uživatelskými aplikacemi a má limitovaný přístup k paměti a k jádru. Ten má povolen pouze přes systémové volání. Kromě jádra zde vše běží. Jádro má za povinnost spravovat všechny uživatelské procesy a aplikace v uživatelském režimu, aby došlo zabránění míchání v režimu jádra. [15]

Jedním z hlavních rozdílů mezi uživatelským režimem a režimem jádra jsou CPU[2] ringy. Ringy jsou vrstvy mezi nainstalovanými aplikacemi na systému a procesy jádra systému. Jsou rovněž jednou ze složek ochrany procesů před rušením. V x86 módu[3] je vždy CPU v jednom ze čtyř ringů. Linux jádro například používá jenom ring 0 a 3. Ring 0 je využíván pro režim jádra a ring 3 je využíván pro uživatelský režim. [15]

[2] CPU je centrální procesorová jednotka. Obsahuje hlavní paměť, kontrolní a aritmeticko-logickou jednotku. [16]

[3] x86 mód je typ architektury CPU. [17]

4 Představení laboratorních úloh

V této kapitole si rozebereme jednotlivé laboratorní úlohy. Každá z nich začíná teorií, která vysvětluje vše potřebné k následující praktické úloze. Ke každé laboratorní úloze je také přidán seznam příkazů pro dané úkony. Na konci každé laboratorní úlohy jsou také vypsány výsledky dané praktické úlohy.

První laboratorní úloha uvedená v kapitole 5 se zabývá instalací a bootováním systému Ubuntu. Dále jsou představeny kroky jako práce s terminálem a vi editorem, operace s balíčky nebo instalace webového prohlížeče. Cílem této kapitoly je vysvětlit uživatelům instalaci Ubuntu a úvodní kroky po ní.

Tématem druhé laboratorní úlohy v kapitole 6 je konfigurace hardwaru, logování a zjišťování informací o CPU a paměti. Dále se zaobírá vytvářením oddílů a využíváním cronu. V této kapitole bude probrána teorie konfigurace systému a hardwaru.

Třetí laboratorní úloha uvedená v kapitole 7 se zabývá konfigurací sítě a zabezpečení systému pomocí radius serveru. Dále je představeno šifrování pomocí GPG klíčů. V této úloze bude výklad ke konfiguraci sítě a zabezpečení systému.

Čtvrtá laboratorní úloha v kapitole 8 se zabývá prací se soubory. Je tu představeno vypisování a určení typů souborů a komprese souborů. Dále je zmíněno hledání, kopírování, přesouvání a odstraňování souborů a adresářů. Cílem této úlohy je představit uživatelům práci se soubory na systému.

Pátá laboratorní úloha uvedená v kapitole 9 se zabývá administrací systému. Je tu představena správa uživatelských účtů a skupin. Cílem je představit správu systému a vysvětlit uživatelům důležitost zabezpečení systému pomocí účtů a skupin.

5 Instalace, bootování a úvodní nastavení systému

Cílem této laboratorní úlohy je vysvětlit boot proces. Bude ukázán podrobný postup instalace distribuce Ubuntu na přenosné médium, včetně tvorby instalačního média. Dále bude představena práce s packet managerem využívaným pro instalaci doplňkového software. Na závěr bude ukázána základní práce s terminálem (CLI).

5.1 *Boot proces*

Boot proces je rozdělen do několika fázi, které jsou níže podrobně představeny a, je-li to nutné či možné, je ukázána jejich parametrizace.

5.1.1 Firmware fáze

V první fázi hardwarové zařízení provede tzv. power on self test (POST), aby byla zajištěna funkce hardware. Po jeho úspěšném dokončení je namapována operační paměť, disková úložiště a případná zapojená USB zařízení. V další fázi je z dostupných zdrojů identifikován bootovací disk a u systémů využívajících BIOS se načte master boot record (MBR) nebo u zařízení využívajících UEFI je nalezen bootloader z ESP partition (EFI System Partition). Zařízení využívají UEFI pak za využití GRUB bootloaderů, nainstalují své EFI verze do ESP (standardně označováno jako GRUB-EFI). [18]

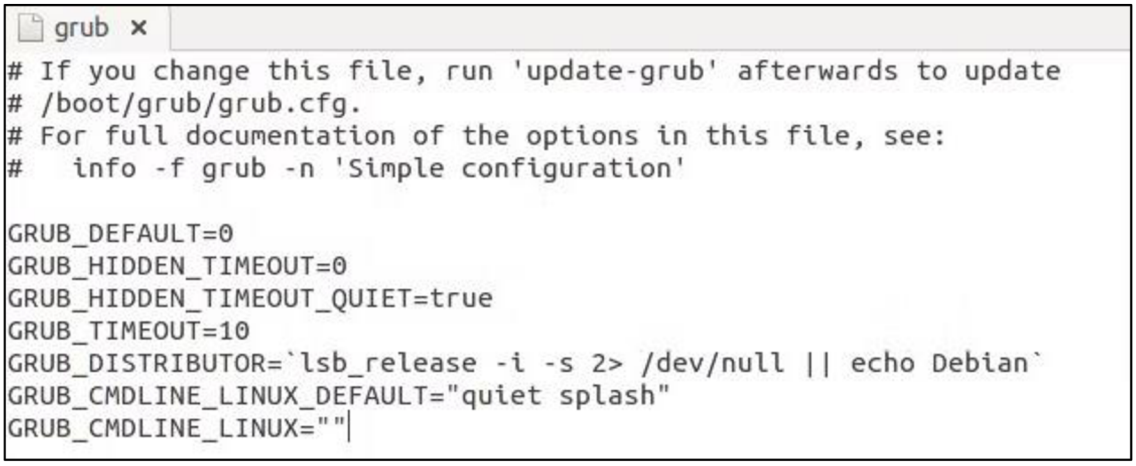
5.1.2 Bootloader fáze

MBR obsahuje první část bootloaderu, jehož úkolem je načíst následující části bootloaderu. Hlavním cílem téhle fáze je načtení jádra (kernel) do operační paměti a následně jej spustit. Dnes je jako bootloader nejběžnější využíván GRUB ve verzi GRUB2. V rámci této fáze je možné vybrat konkrétní operační systém, nalezl-li bootloader více verzí/typů (dual či multi-boot). Hlavním úkolem bootloader fáze je načtení a spuštění kernelu, kterému v posledním kroku předává kontrolu nad následujícími fázemi procesu.

Bootloadery

V této podkapitole budou představeny GRUB soubory a práce s nimi. Některé z klíčových GRUB souborů jsou:

- **grub.cfg** je konfigurační soubor GRUB a obsahuje informace o načítání operačního systému a konfiguraci GRUB. Konfigurace může obsahovat informace o dostupných jádrech systému, jejich parametrech a dalších volbách.
- **grubenv** je složen z informacích o prostředí GRUB. Může také obsahovat proměnné a další konfigurační údaje ovlivňující GRUB. unicode.pf2 je soubor s písmovým fontem, který je používán GRUBem pro zobrazení textu na obrazovce.
- GRUB také může obsahovat soubory **stage1**, **stage2** nebo **stage1.5**, což jsou části GRUB nacházející se na samotném začátku disku a jsou zodpovědné za zavedení GRUB do paměti a k načítání dalších částí GRUB. V GRUB2 jsou stage1 a stage2 nahrazeny jediným souborem, který obsahuje obě funkce.



Obrázek 1: ukázka grub.cfg souboru [19]

Pro zobrazení **grub.cfg** souboru využijeme příkazu **cat/boot/grub/grub.cfg**. Pokud na začátek přidáme argument head (namísto cat), uvidíme jen část tohoto souboru. Pro jeho editaci využijeme příkazu **vi /etc/default/grub**. V tomto editoru můžeme například změnit grub timeout nastavení udávající čas, po který systém čeká na input z klávesnice než se nabootuje výchozí menu entry. Pokud zde nastavíme 0, ihned se vybere výchozí entry bez zobrazení menu, pokud nastavíme -1, bude systém čekat na první input.

Recovery mode – praktické využití grub.cfg

Pokud chceme vidět, jak vypadá GRUB menu, můžeme se tam dostat přes recovery mód. Recovery mód obvykle využíváme, pokud má náš systém problém s bootováním. Abychom se dostali do recovery módu, musíme při bootování systému stisknout klávesu **Esc** před tím, než odpočet dosáhne 0. Pokud naše zařízení používá UEFI fast boot, tak toto bootování může být až tak rychlé, že nám ani nedá možnost stisknout jakoukoliv klávesu. [20] Toto nás dostane do GRUB, kde se pohybujeme pomocí šipek a Enteru. Dále zvolíme pokročilé možnosti, a tam vybereme možnost bootování do recovery módu. Vybereme editační mód stisknutím klávesy **E** namísto Enteru. Takto uvidíme recovery mód.

5.1.3 Kernel fáze

Úkolem kernelu je inicializace ovladačů hardware a napojení kořenového souborového systému. Úvodní komponenty kernelu jsou k němu staticky napojeny, jelikož sdílené knihovny nejsou v této fázi dostupné. Kód kernelu je z důvodů jeho velikosti zabalený, tzv. komprimovaný, dnes nejčastěji za využití bzImage a v rámci této fáze realizuje své rozbalení do operační paměti. bzImage vychází z komprese pomocí bzip2, což je jeden z možných typů komprese používaných pro tento typ obrazu. bzImage obsahuje komprimovaný obraz jádra a kód pro extrakci a spuštění jádra. Následně tento obraz může být použit bootloaderem (jako např. GRUB) k načtení jádra do paměti a spuštění operačního systému. Název bzImage se obvykle používá ve spojení s Linuxem, zatímco v jiných operačních systémech mohou existovat odlišné formáty obrazů jádra. Některé ovladače, které zatím nejsou podporovány kernelem, jsou uloženy na initrd[4] disku. Po svém rozbalení se kernel

[4] Initrd (initial RAM disc) je dočasný systémový oddíl, který je dočasně namontován do paměti během bootování OS. Jeho hlavním cílem je umožnit operačnímu systému spustit se na hardwaru bez využití specifických ovladačů, které nejsou součástí jádra. Initrd je v podstatě archiv souborů a skript potřebných pro inicializaci a načtení jádra do paměti. Pro představu, initrd se zavádí do paměti a inicializuje během bootloader fáze. Po jeho inicializaci se načte kernel, který pokračuje ve spouštění počítače.
Mezi důležitými prvky, které obsahuje jsou například:

spustí, namapuje initrd a opětovně namapuje i reálný kořenový souborový systém. Na závěr této fáze kernel vytvoří a spustí proces init, tj. první proces, který je součástí systému až do jeho vypnutí.

5.1.4 Init fáze

Jedná se o finální fázi bootování systému. Proces init má za úkol zprovoznit nastavené služby, zobrazit login obrazovky a nastavit konzoli. Dále má za úkol zavedení a spuštění všech ostatních systémových procesů[5]. V moderních distribucích je proces init nejčastěji nahrazován procesem systemd.

Init je tradičním systémem pro inicializaci systémů Unix/Linux a je známý svou jednoduchostí, což v dnešních složitějších systémech bývá spíše nevýhodou. Historicky byl používán ve většině distribucí. Jeho konfigurace definuje, jak jsou spouštěny služby a procesy, a je obvykle obsažena v jednom nebo více konfiguračních souborech. Mezi nimi může být například **/etc/inittab**. Jako alternativa k init byl vyvinut systemd, jehož cílem bylo zvýšit efektivitu a rychlost inicializačního procesu. Toho je dosaženo pomocí paralelní inicializace služeb.

Systemd rovněž není pouze inicializační systém, ale i systém pro správu procesů a služeb. Dále také obsahuje například nástroje pro sledování a logování či správu uživatelských sessions. Jeho konfigurace je obvykle prováděna pomocí service units a dalších konfiguračních souborů. Zde také stojí za zmínku, že i když je init nahrazován Upstart či systemd, jeho náhrady používají spustitelné soubory pojmenované init, a to z důvodů kompatibility s legacy procesy.

Pokud kernel vyžaduje ovladače disku, které jsou však umístěny na něm samotném, je využíván **initramfs**[6] **(Initial RAM Filesystem)**, což je počáteční kořenový souborový systém.

Moduly ovladačů, které jsou nezbytné pro inicializaci a podporu hardwaru během bootloader fáze. Tyto ovladače můžou být dynamicky načteny do paměti, díky čemu může jádro pracovat s různým hardwarem.

Skripty a nástroje pro inicializaci, které mají za úkol připravit systém pro předání kontroly kernelu.

[5] Pokud se některý z procesů odpojí od svého rodičovského procesu, tak ho init musí adoptovat.

[6] Initramfs je dočasný souborový systém, který je načten do paměti během init procesu OS. Hlavním účelem initramfs je poskytnout základní prostředí a nástroje, které jsou nezbytné pro spuštění a

5.1.5 Runlevels

V systémech založených na Unixu, tedy včetně mnoha linuxových distribucí, jsou využívány tzv. runlevels (úrovně běhu), které identifikují specifický stav operačního systému. Každá úroveň reprezentuje určitý stav systému při jeho spouštění nebo vypínání. Systém může být v různých runlevels v závislosti na konfiguraci konkrétních služeb a procesů. Základní runlevels jsou standardně označovány čísly od 0 do 6, přičemž každá hodnota odpovídá určitému stavu systému.

- Runlevel 0: Vypnutí systému.
- Runlevel 1 nebo S: Single-user režim, často používaný pro údržbu nebo opravy.
- Runlevel 2: Multi-user režim bez síťových služeb (někdy vzdálený přístup pouze přes textový terminál).
- Runlevel 3: Multi-user režim s textovým přihlášením a všechny běžící služby.
- Runlevel 4: Někdy je volný pro uživatelskou konfiguraci (význam může záviset na konkrétní distribuci).
- Runlevel 5: Multi-user režim s grafickým přihlášením (X Window System).
- Runlevel 6: Restart systému.

V modernějších distribucích operačního systému Linux, zejména těch, které používají systemd, je koncept runlevelů nahrazen nebo doplněn jinými mechanismy. Systemd pro organizaci různých stavů systému místo runlevelů používá tzv. "target units". Každá target unit může definovat sadu služeb, které mají být spuštěny nebo zastaveny v daném stavu systému. Tyto jednotky mohou být využívány pro organizaci různých scénářů jako například start systému bez grafického uživatelského rozhraní, spuštění v režimu jednoho uživatele, nebo aktivaci síťových služeb.

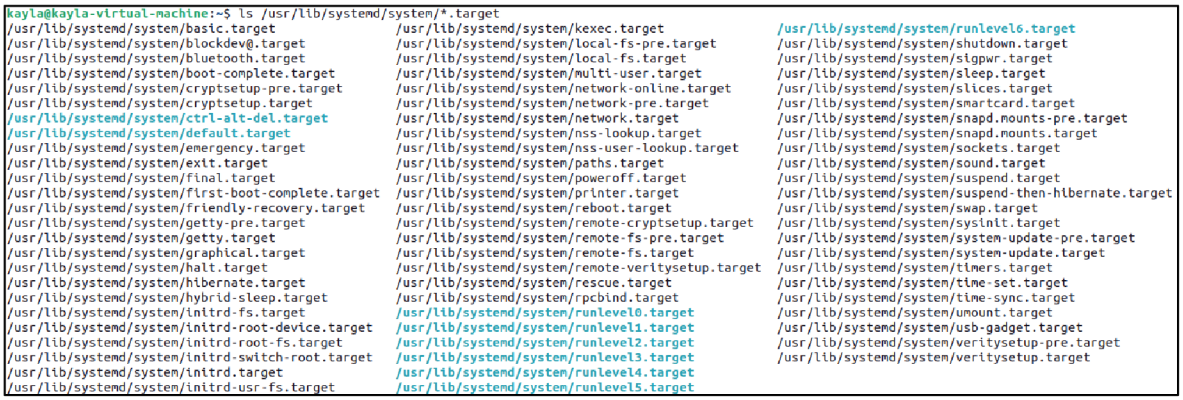
Několik běžných target units v systemd zahrnuje: [21]

---

načtení kernelu. Kromě modulů, ovladačů a skript pro konfiguraci, také například obsahuje nástroje pro montáž a inicializaci kořenového souborového systému, detekci hardwaru.
Rozdíl mezi initramfs a initrd je takový, že initramfs je obvykle archiv obsahující komprimovaný souborový systém oproti initrd, který může být buď archiv nebo obraz disku.

- multi-user.target: Podobné tradičnímu runlevelu 3, spouští multi-user režim s textovým přihlášením a všechny běžící služby.
- graphical.target: Podobné tradičnímu runlevelu 5, spouští multi-user režim s grafickým přihlášením (X Window System).
- rescue.target: Podobné tradičnímu runlevelu 1 nebo S (single user mode), slouží pro single user režim, obvykle pro údržbu nebo opravy.
- emergency.target: Nastavuje systém do nejmenšího možného running mode pro záchranné účely, často bez síťového připojení.
- poweroff.target: Ukončuje systém a vypíná ho.
- reboot.target: Restartuje systém.

Jednotky v systemd jsou definovány konfiguračními soubory s příponou .target v adresáři /etc/systemd/system/ nebo /usr/lib/systemd/system/. Použitím target units systemd umožňuje efektivnější správu a organizaci běhových stavů systému a poskytuje flexibilnější konfiguraci než tradiční init systémy.



Obrázek 2: soubory s příponou .target v adresáři /usr/lib/systemd/system/ (zdroj: autor)

A nyní trochu prakticky. Pomocí příkazu **ls /etc/init.d | more** si můžeme zobrazit jaké služby konkrétní systém využívá. Pro zobrazení runlevelů určité služby využijeme příkazu **grep runlevel (cesta k souboru)**, například: **grep runlevel /etc/init.d/ssh**. Dále také můžeme vyzkoušet restartování služby pomocí **/etc/init.d/ssh restart**.

5.1.6 Kernel zprávy

Abychom mohli vidět zprávy generované kernelem při bootování, využijeme příkazu **dmesg,** který zobrazuje tzv. ring buffer[7], což může být užitečné při troubleshootingu, sledování inicializačních procesů nebo zjišťování chyb systému. Při spuštění tohoto příkazu uvidíme seznam zpráv jádra, které byly vygenerovány od posledního startu systému. Zprávy související se systémem jsou uloženy v **/var/log/messages** souboru. Tradičně ho updatují **syslogd** a **klogd** démoni. Na systemd systémech je to binární soubor a updatuje ho **journald** démon. **Journald** soubory lze prohlížet pomocí **journalctl**[8] příkazu a output je podobný **dmesg** příkazu.

5.2 *Instalace Ubuntu*

1. V této kapitole si ukážeme, jak nainstalovat Ubuntu na své Windows zařízení.[22] Ubuntu image stáhneme z https://ubuntu.com/download/desktop
2. Vytvoříme si bootovatelné USB. Lze využít third-party aplikací nebo těchto kroků na Windows zařízení:
    a) Otevřeme cmd nebo PowerShell jako administrátoři. Využijeme příkazu **diskpart,** a poté **list disk**, kde najdeme své USB.
    b) Příkazem **select disk X**, kde X je číslo našeho USB, vybereme své USB.
    c) Poté vymažeme všechna data na USB pomocí příkazu **clean** a vytvoříme primary partition[9](viz kapitola 6.1.3 Vytváření oddílů) díky příkazu **create partition primary**.
    d) Vybereme typ oddílu **select partition 1.**
    e) Zformátujeme oddíl na NTFS (**format fs=ntfs quick**) nebo FAT32(**format fs=fat32 quick**).
    f) Nastavíme oddíl jako aktivní: **active.**

---

[7] Ring buffer je místo, ve kterém kernel uchovává různé zprávy a informace během svého běhu. Data v ring bufferu obsahují informace o inicializaci hardwaru, detekci zařízení, chybách a dalších relevantních událostech ohledně jádra systému.
[8] Malé L.
[9] Primary partition je oddíl disku, na který lze nainstalovat operační systém.

g) Přiřadíme disku jeho label automaticky pomocí příkazu **assign** nebo můžeme vybrat určité písmeno **assign letter= X**, kde X písmeno našeho výběru, vyjma písmen, která již byla použita.

h) Ubuntu image extrahujeme z ISO souboru, a poté zkopírujeme tyto soubory do našeho USB.

3. V tomto kroku se musíme dostat do BIOSU/UEFI nastavení a mít při tom USB v počítači. Do BIOSU se můžeme dostat přes *Nastavení* – spuštění například stisknutím *Windows klávesy + I* – a poté v sekci Systém a podsekci Obnovení vybereme *Spuštění s upřesněným nastavením*. Toto nám automaticky restartuje zařízení. Po restartu vybereme možnost *Troubleshoot*. Dále zvolíme variantu *Advanced Options*, tzv. pokročilé možnosti, a poté *UEFI Firmware Setting*s nebo *BIOS Settings*. V tomto nastavení se může stát, že nebude fungovat myš, tudíž bude potřeba se orientovat šipkami.
4. V BIOS/UEFI vybereme *Boot* nebo *Boot Order* sekci. Může být pojmenována i jinak, záleží na daném systému. V této sekci nastavíme nejprve bootování Ubuntu. Toto nastavení je pro nás lepší variantou, jelikož se tím snižuje šance přepsání GRUB souborů Windows aktualizacemi. Pokud nám Windows alespoň částečně přepíše GRUB soubory, zabere nám poměrně dost času je opravit.
5. Uložíme změny, opustíme BIOS/UEFI nastavení a restartujeme zařízení. Při bootu by se systém měl načítat z našeho USB. Dále by se měl otevřít instalační program, který nás provede dalším nastavením.
6. Vybereme si jazyk, ve kterém chceme Ubuntu používat.
7. V dalším kroku nám program nabídne možnost instalace Ubuntu nebo pouhého vyzkoušení systému. Vybereme možnost instalace.
8. Dále vybereme rozložení klávesnice a připojíme se k Wi-Fi.
9. V dalších krocích nás může instalační program vyzvat k vypnutí některých funkcí (např. Intel RTS), které Ubuntu nepoužívá.
10. Pro instalaci si můžeme vybrat mezi normální a minimální instalací, a také si můžeme nechat nainstalovat některé third-party programy.
11. Je možné nainstalovat Ubuntu podél současně používaného operačního systému nebo si jej nainstalovat jako jediný používaný operační systém.

Pokud si jej nainstalujeme zároveň s jiným operačním systémem, budeme muset vybrat, na který disk a oddíl bude nainstalován a kolik místa bude mít přiděleno.

12. Dále máme možnost si zašifrovat celý disk pomocí LVM a LUKS. K této možnosti je třeba být důkladně obeznámen s problematikou těchto šifrování, v opačném případě se tento postup nedoporučuje[10].
    i. LVM je metoda správy logických disků, také je to jeden ze způsobů virtualizace diskového prostoru.
    ii. LUKS je software pro šifrování pevných disků.[11]
    b) Bude třeba si vybrat bezpečnostní klíč, který bude při bootování zadán.
    c) Mezi další možnosti také patří manuální rozdělení disku do oddílů.
13. Pokud instalujeme Ubuntu na zařízení, které používá operační systém Windows a chceme si oba operační systémy ponechat, je třeba mít vypnutý Windows BitLocker, jinak Ubuntu nebude moci získat informace o dostupných discích. Při úplném smazání Windows toto není třeba řešit.
14. V dalším kroku uvidíme souhrn naší vybrané konfigurace, kterou ještě můžeme upravit.
15. Dále vybereme polohu a časové pásmo, vytvoříme root login s user loginem a dokončíme instalace.
    a) Je žádoucí vytvořit root účet i user účet s jinými hesly. Root účet je admin účet s největšími oprávněními, a proto používání tohoto účtu jako hlavního uživatelského účtu vytváří bezpečnostní rizika. Tímto také zamezíme možnosti omylem změnit důležitá nastavení. Pokud chceme, aby náš user účet měl možnost provádění úkonů s právem roota, můžeme nastavit možnost použití *sudo* přístupu. *Sudo* příkaz – superuser do – nás na dobu určitou změní na účet s root právy, ale

---

[10] How to full encrypt your linux system with lvm on luks - Linux.com

[11] How to full encrypt your system with lvm on luks from cli - Linux.com

narozdíl od používání root účtu po provedení příkazu bude náš účet mít opět jen uživatelská oprávnění. Při využívání *sudo* příkazu budeme dotázáni na naše user heslo pro potvrzení oprávnění k využití *sudo* příkazu.

16. Lze si ještě vybrat mezi světlým a tmavým motivem a poté budeme požádáni o restart zařízení.
17. Po restartu vysuneme USB, zadáme uživatelské heslo a případně šifrovací heslo, čímž dokončíme instalaci.
18. Měli bychom také zkontrolovat aktualizace systému. Přestože nainstalujeme nejnovější verzi Ubuntu, může se stát, že nebudou nainstalovány všechny aktualizace.

5.3 *Práce s terminálem*

Terminál patří mezi nejdůležitější způsoby práce se systémem Ubuntu. Ve Windows s terminálem pracujeme jen ojediněle, zato v Linux distribucích se s terminálem pracuje na každodenní bázi. Proto si ukážeme základy práce s terminálem.

Terminál můžeme otevřít použitím klávesové zkratky **Ctrl + Alt + T** nebo pomocí hledání „Terminal“ či „Command“ v Menu aplikací.

V této tabulce jsou vypsány základní příkazy a jejich funkce.

Tabulka 1: příkazy terminálu a jejich funkce (zdroj:autor)

ls	Výpis souborů v aktuálním adresáři
cd xxx	Změna adresáře na xxx
pwd	Vypsání celé cesty (celého jména) k adresáři
cp xxx yyy	Zkopírování souboru nebo adresáře xxx do adresáře yyy
mv xxx yyy	Přesunutí či přejmenování souboru xxx do yyy. Přejmenování souboru je bráno jako přesunutí souboru xxx do nového souboru yyy
rm xxx	Smazání souboru nebo adresáře xxx
mkdir xxx	Vytvoření nového adresáře xxx
man xxx	Zobrazení stránky s manuálem pro daný příkaz

..	Využíváno pro odkazování na nadřazený adresář (parent directory)
.	Využíváno pro odkazování na aktuální adresář
uname *-a*	Vypsání informací o systému. Lze použít s různými možnostmi, např. *-a* vypíše všechny dostupné informace.
vi xxx	Otevření vi textového editoru pro soubor xxx
history	Zobrazení historie příkazů

Mezi další užitečné klávesové zkratky můžou patřit **Ctrl + L**, jež smaže data z obrazovky terminálu, nebo **Ctrl + C**, která přeruší právě běžící příkaz.
V neposlední řadě nezapomeňme, že příkazy v terminálu jsou citlivé na velká a malá písmena. Většina příkazů je napsaná pomocí malých písmen.

5.4 *Instalace webového prohlížeče*

V této kapitole si ukážeme, jak nainstalovat aplikace na příkladu webových prohlížečů, jelikož se může jednat o jednu z prvních činností, co uživatelé v novém operačním systému udělají. Existují různé přístupy ke stažení webových prohlížečů. Většina prohlížečů může být nainstalována pomocí *APT* příkazů, ale ukážeme si i jiné varianty instalace.
První příklad bude instalace Google Chrome z webových stránek, což je asi nejčastější způsob. [23]

1. Na stránce https://www.google.com/chrome/ vybereme stažení 64bitového .deb souboru pro Debian/Ubuntu.
2. Soubor bychom měli později najít ve Stažených souborech a po jeho otevření se otevře Ubuntu Software aplikace, která nám dá možnost instalace.
3. Bude třeba zadat naše heslo, abychom potvrdili souhlas s instalací aplikace z neznámého zdroje.

Druhým způsobem je instalace Firefoxu a Google Chrome z terminálu.

1. K instalaci Firefoxu nám stačí jediný příkaz, a to: **sudo apt-get install firefox**.

2. Pokud bychom chtěli nainstalovat Google Chrome, postupujeme podle následujících kroků:
    a. Nejdříve využijeme příkazu **wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb**
        i. Tento příkaz stáhne daný *.deb* soubor.
    b. Po stažení použijeme příkaz **sudo dpkg -i google-chrome-stable_current_amd64.deb**. Budeme dotázáni na naše heslo.
    c. Poslední příkaz bude **sudo apt-get install -f**.

Třetím způsobem může být v některých případech instalace pomocí Snap packages. Nainstalovat takto můžeme například Chromium nebo Firefox. Ukážeme si instalaci Chromium.

1. Znovu stačí pouze jediný příkaz **sudo snap install chromium**.

Například prohlížeč Opera lze nainstalovat všemi předešlými způsoby, ale i pomocí Ubuntu Software Manageru. [24]

1. Otevřeme Ubuntu Software Manager z Panelu aktivit a do vyhledávače zadáme *opera* nebo *opera-beta*.
2. Klikneme na tlačítko instalace a po proběhnutí instalace můžeme přejít do Panelu aktivit a hledat Operu. Poté stačí kliknout na ikonu Opery a otevře se nám prohlížeč.

5.5 *Odinstalace webového prohlížeče*

V této kapitole si ukážeme, jak odinstalovat webový prohlížeč a pro příklad použijeme Firefox. Zde si ukážeme, jak odinstalovat aplikace v terminálu.

1. Otevřeme terminál.
2. Použijeme příkaz **sudo apt-get remove firefox,** pokud používáme účet s oprávněním sudo. Pokud máme root účet nebo účet se superuser oprávněním, použijeme příkaz **apt-get remove firefox**.
    a. Pokud neznáme přesný název balíčku, lze využít příkazu **dpkg -l | grep name**, kde místo **name** použijeme název aplikace či zkusíme najít jméno v Package manageru.

3. V dalším kroku můžeme odinstalovat konfigurační soubory. Využijeme příkazu **apt-get purge firefox** nebo **sudo apt-get purge firefox**.
4. Po odinstalování Firefoxu aktualizujeme seznam balíčků pomocí příkazu **apt-get update**.

Využili jsme příkazů *remove* a *purge*, které mohou vypadat podobně. Příkaz *remove* pouze odinstaluje balíček aplikace, ale zachová konfigurační soubory pro případ opětovného nainstalování aplikace. Příkaz *purge* smaže balíček aplikace i s konfiguračními soubory. Smazání aplikace také smaže všechna nastavení a data aplikace, takže je třeba si zkontrolovat, že odinstalujeme správnou aplikaci a že máme všechna důležitá data uložena.

5.6 *Další operace s balíčky*

Mezi další příkazy **apt** patří **apt-get upgrade**, který nainstaluje nejnovější verze všech balíčků. Výstup tohoto příkazu nabídne seznam balíčků, jež byly aktualizované, nově nainstalované či smazané. Dále vám může nabídnout balíčky, které již nejsou potřeba a které bychom mohli smazat.
Pokud chceme vidět seznam repositářů, ze kterých lze nainstalovat aplikace, použijeme příkaz **tail /etc/apt/source.list**.

5.7 *Práce s vi editorem*

Vi editor je jeden z nejpoužívanějších textových editorů v Unixových operačních systémech, jenž má dva módy, ve kterých lze pracovat. Prvním z módů je příkazový mód. V tomto módu můžeme vytvořit, přejmenovat či uložit soubor. Dále můžeme například kopírovat a vkládat soubor. Jinými slovy, lze provádět téměř všechny úkony vyjma úpravy textu v souboru. Druhý mód je tzv. Insert mód, ve kterém můžeme vkládat a upravovat text v souboru. První mód je výchozí, pro vstup do druhého módu je třeba zadat příkaz (viz níže). [25]
Používání vi editoru zahájíme příkazem **vi [název souboru]**. Pro vytvoření souboru stačí zadat název souboru, který zatím neexistuje, čímž se soubor vytvoří. Pro editaci zadáme název souboru, který chceme editovat. Při zadání **vi** příkazu se defaultně nacházíme v příkazovém módu. V tomto módu má většina kláves svoji funkci, kterou je třeba potvrdit stisknutím klávesy **Enter**. Mezi často používané příkazy patří **:wq**

pro přepsání a zavření souboru s uložením provedených změn. Pro uzavření souboru bez uložení změn použijeme **:q**. [25]
Pro vstup do insert módu stiskneme klávesu **i**. V tomto módu se pohybujeme pomocí šipek. Pro výstup z insert módu stiskneme klávesu **Esc**. Poté je třeba uložit změny a zavřít soubor či jen zavřít soubor. [25]

5.8 *Man pages*

Man pages jsou soubory s manuálem, které pomáhají uživateli s používáním a vysvětlením příkazu. Pro zobrazení man page příkazu využijeme příkazu **man [název příkazu]**. Struktura man page obsahuje tyto sekce: název příkazu (name), syntax příkazu (synopsis), popis (description), možnosti příkazu (options), autor (author), pokyny k hlášení chyb vývojářům (reporting bugs), informace o copyrightu (copyright), informace o plné dokumentaci či podobných příkazech (see also), konfigurační detaily příkazu (configuration) a možnosti hodnot výstupního statusu (exit status). Každá man page může mít tuto strukturu, některé však neobsahují všechny uvedené sekce. Skoro všechny obsahují aspoň sekce název příkazu, syntax příkazu a možnosti příkazu. Tyto stránky lze procházet pomocí **mezerníku** po stránkách či pomocí **šipek** na klávesnici po řádcích. Pro opuštění man pages využijeme klávesy **q**. [26]

5.9 *Praktická úloha*

V této praktické úloze si nainstalujeme aplikaci Docker. Nejdříve zkuste praktickou úlohu provést sami bez návodu.

Kroky postupu jsou:

1. Upravení *apt* repozitáře + přidání oficiálního GPG klíče pro Docker
2. Přidání repozitáře Docker do *apt* zdrojů
3. Instalace Docker
4. Spuštění **hello-world** image

5.9.1 Návod

Postup je z oficiální dokumentace aplikace Docker. [27]

Stažení aplikace Docker

Nejdříve nastavíme *apt* repozitář. Pro správné nastavení si nejdříve přidáme oficiální GPG klíč pro Docker:

1. **sudo apt-get update**
2. **sudo apt-get install ca-certificates curl**
3. **sudo install -m 0755 -d /etc/apt/keyrings**
4. **sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc**
5. **sudo chmod a+r /etc/apt/keyrings/docker.asc**

Poté přidáme Docker repozitář do apt zdrojů:

1. **etc \ "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \ $(. /etc/os-release && echo "$VERSION_CODENAME") stable " | **
2. **sudo tee /etc/apt/sources.list.d/docker.list > /dev/null**
3. **sudo apt-get update**

Nainstalujeme Docker složky pomocí příkazu **sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin**.
A po instalaci si zkontrolujeme, že instalace proběhla správně pomocí spuštění **hello-world** image (**sudo docker run hello-world**). Tento příkaz stáhne testovací image, spustí ho a poté vypíše potvrzovací zprávu a zavře se.

5.10 *Otázky na ověření znalostí*

1. Jak funguje proces bootování systému?
   a) Proces bootování je rozdělen do čtyřech fází: firmware, bootloader, kernel a init. Ve firmware fázi systém zkontroluje správnou funkci hardware a načte se MBR pro BIOS systémy nebo ESP partition pro UEFI systémy. UEFI systémy pak pomocí GRUB bootloaderů nainstalují EFI verze do ESP. V Bootloader fázi se načte kernel do operační paměti a spustí se. Je k tomu využíván GRUB2 bootloader. Cílem kernel fáze je inicializace ovladačů hardwaru a napojení kořenového souborového systému. Nejdříve se musí rozbalit bzImage kernelu do operační paměti. Poté je tento image použit bootloaderem k načtení jádra a spuštění operačního systému. Dále kernel namapuje initrd i reálný souborový systém. V posledním kroku vytvoří a spustí init proces. V init fázi se zprovozní nastavené služby a zavedou a spustí se ostatní systémové procesy. Proces init je dnes nahrazován procesem systemd, který je rychlejší a efektivnější díky paralelní inicializaci služeb.
2. Jak poznat runlevely a jak jsou rozděleny?
   a) Runlevely identifikují specifický stav operačního systému. Runlevely jsou rozděleny číselně od 0 do 6. Mezi příklady runlevelů patří : 0 pro vypnutí systému; 1 často používaný na údržbu a opravy a 6 odpovídá restartu systému.
3. Jaký je rozdíl mezi runlevely a target units?
   a) Runlevely jsou konceptově jednodušší, jelikož reprezentují již definovaný stav systému. Target units jsou více flexibilní a oproti runlevelům jsou výstižněji pojmenované.
4. Proč jsou důležité kernel zprávy?
   a) Kernel zprávy mohou být užitečné při troubleshootingu, jelikož popisují všechno, co je prováděno na systému.
5. K čemu jsou využívány man pages?

a) Man pages jsou využívány pro nápovědu k používání příkazům. Tyto stránky by měli obsahovat všechny dostupné informace, které uživatel potřebuje k využití daného příkazu.

6 Konfigurace systému a hardwaru

Cílem této laboratorní úlohy je vysvětlit, jak zkontrolovat, případně opravit konfiguraci hardwaru. Dále bude vysvětleno, jak vytvořit oddíly, tzv. partitions, změnit souborový systém a vytvořit swap soubory.

6.1 *Konfigurace hardwaru*

6.1.1 Informace o CPU

V této kapitole si ukážeme, jak si zobrazit informace o CPU, zaplnění paměti a o připojených USB zařízeních.

Pokud si chceme zobrazit nejdůležitější informace o CPU, využijeme příkazu **lscpu**.

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) per core:    2
Core(s) per socket:    2
Socket(s):             2
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            15
Model:                 4
Stepping:              8
CPU MHz:               2800.000
BogoMIPS:              5585.82
L1d cache:             16K
L2 cache:              2048K
NUMA node0 CPU(s):     0-7
```

Obrázek 3: ukázka výstupu příkazu lscpu [28]

Tento příkaz vypíše například architekturu CPU, počet CPU či kolik máme vláken na jádře. Dále můžeme vidět i výrobce, model a jeho název či sokety a kolik jader v nich máme. Dále k tomuto příkazu můžeme přidat argument -e (lscpu -e) a příkaz bude v tzv. lidmi čitelném formátu. K tomuto argumentu dále můžeme doplnit názvy sloupců, které chceme zobrazit, a tím si nechat zobrazit jen důležité sloupce. Pokud bychom využili argumentu -p(lspcu -p), zobrazí se nám informace v tvaru vhodném pro zpracování, jenž můžeme vložit do jiných programů pro pomoc s analýzou problému. Dále také můžeme příkaz použít s argumentem -J¬ (lscpu -J), který nám výsledek zobrazí v JSON formátu. [29]

Pro detailnější informace o CPU si můžeme zobrazit soubor **/proc/cpuinfo** pomocí příkazu **cat /proc/cpuinfo**.

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 94
model name      : Intel(R) Celeron(R) CPU G3900 @ 2.80GHz
stepping        : 3
microcode       : 0xea
cpu MHz         : 899.999
cache size      : 2048 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi
mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good
nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg cx16
xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave rdrand lahf_lm abm
3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid
ept_ad fsgsbase tsc_adjust erms invpcid rdseed smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves
dtherm arat pln pts hwp hwp_notify hwp_act_window hwp_epp md_clear flush_l1d
vmx flags       : vnmi preemption_timer invvpid ept_x_only ept_ad ept_1gb flexpriority tsc_offset vtpr mtf
vapic ept vpid unrestricted_guest ple pml
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs itlb_multihit srbds
```

Obrázek 4: ukázka části výstupu příkazu cat /proc/cpuinfo [30]

Zde ve výstupu můžeme vidět například počet procesorů, rychlost konkrétního procesoru v MHz, velikost cache a vlajky. [31] Vlajky jsou seznam vlastností a možností CPU. Různé vlajky označují různé specifické sety instrukcí a rozšíření, které CPU podporuje. Toto nám může pomoci při určení typu softwarové optimalizace na daném systému. [30]

6.1.2 Informace o paměti

V této kapitole si ukážeme, jaké příkazy máme použít pro zjištění informací o paměti. Toto bychom měli využít, pokud máme problémy s rychlostí zařízení nebo při výměně hardwaru či při optimalizování výkonnosti zařízení. Využijeme k tomu příkazu **free**. Tento příkaz nám umožní zjistit jaká část paměti RAM je přístupná pro systém. Ve výstupu se zobrazí tabulka paměti RAM s hodnotami celkové, zaplněné a

volné kapacity. Dále můžeme vidět sdílenou, dostupnou[12] a cache paměť[13]. Jsou zobrazeny i statistiky pro swap paměť, podobné tabulce hodnot paměti RAM. [32]
Dále můžeme využít virtuálního souboru /proc/meminfo pro zobrazení více informací o paměti. Příkazem **cat /proc/meminfo** zobrazíme tento soubor. Výstup lze rozdělit do několika kategorií. [33]
Obecná paměť: Při použití příkazu **cat /proc/meminfo | grep "Mem"** uvidíme obecné informace o paměti včetně kapacity. [33]
Buffery a cache paměti: Při použití příkazu **cat /proc/meminfo | grep -e "Buffers" -we "Cached"** se zobrazí informace o paměti využívané buffery[14] a cache pamětí.[33]
Swap space: Swap space je záložní místo pro RAM na disku. Příkazem **cat /proc/meminfo | grep "Swap"** zobrazíme kapacitu swapové mezipaměti, celkové swap paměti a volné swap paměti. Seznam různých dostupných swap pamětí je v souboru /proc/swap. [33]
Mezi dalšími body ve výstupu příkazu **cat /proc/meminfo** můžeme také nalézt informace o aktivní či neaktivní paměti, writeback paměti či sdílené paměti. [33]

6.1.3 Vytváření oddílů

Návod v této kapitole je použit z článku Create a Partition in Linux. [34] Rozdělení disku do oddílů může být výhodné pro organizaci souborů či bezpečnosti. Rozdělení disku na oddíly je možností prevence ztráty dat v případě, že by jeden oddíl byl poškozený. V případě bootování vícero operačních systémů je rozdělení disku většinou vyžadováno danými operačními systémy. Nejlepší je provést rozdělení

---

[12] Dostupná paměť není synonymem pro volnou paměť. Volná paměť je v daném okamžiku zcela nevyužitá. Nesmí být využívána ani pro cache či buffery, tudíž je její hodnota na moderních operačních systémech poměrně malá. Dostupná paměť zahrnuje nejen volnou paměť, ale také paměť aktuálně využívanou buffery a cache, která může být snadno uvolněna. Dostupná paměť proto poskytuje lepší pohled na to, kolik paměti je opravdu dostupné pro nové úlohy, aniž by došlo ke zpomalení systému kvůli swapování tzv. přesunu dat na disk.

[13] Cache paměť je mezipamět pro soubory načtené z disku, dále zahrnuje tmpfs (dočasný souborový systém) a shmem(sdílená paměť), ale nezahrnuje SwapCached, což nám značí, kolik paměti ve swapu je momentálně tzv. cachováno.

[14] Buffery jsou dočasný prvek, který obecně nepřesahuje 20 MB.

disků při instalaci operačního systému. Během procesu rozdělení disků po instalaci pracujeme s pamětí, která již může obsahovat data, jež systém přepíše. Rozdělení disku při instalaci jsme si již ukázali v první laboratorní úloze, a proto si v této kapitole ukážeme, jak rozdělit disk po instalaci pomocí příkazu **gdisk**, jelikož pracujeme s GPT[15] disky. Jiná zařízení také mohou využít příkazu **fdisk** nebo **parted**.

Před využitím příkazu **gdisk** či příkazu jemu podobného zvažme, zda potřebujeme náš disk rozdělit, dále zhodnotíme rizika ztráty dat, které máme na svém zařízení, a vše důležité si předem zálohujeme.

Nejdříve si zkontrolujeme oddíly na všech discích pro vybrání správného disku k rozdělení. Toto provedeme pomocí příkazu **fdisk -l [cesta k disku]**. Bude třeba využít sudo privilegií a budeme muset pro kontrolu zadat heslo. Ve výstupu tohoto příkazu uvidíme každý disk, který můžeme využívat, a jeho základní informace, tedy velikost, geometrii disku či jeho ID. Pro každý oddíl vidíme jeho cestu, zda je tzv. bootable[16] a které cylindry oddílu jsou počáteční a konečné. Dále je zobrazena velikost oddílu v blocích, ID oddílu a typ souborového systému či jeho použití.

Po vybrání disku použijeme příkaz **sudo gdisk [cesta disku]**. Ve výstupu tohoto příkazu uvidíme informace o tom, jak postupovat v daném příkazovém módu. Všechny změny budou uloženy v paměti, dokud se nerozhodneme je zapsat a uložit. Před uložením a zapsáním změn si zkontrolujeme nastavení. Příkazový mód může být kvůli svému rozhraní zpočátku matoucí. Tento mód používá jednoznakové příkazy, jejichž seznam lze zobrazit stisknutím klávesy **m**.

Pro vytvoření oddílu využijeme příkazu **n**. Poté budeme vyzváni k zadání čísla oddílu nebo stisknutím klávesy **Enter** můžeme potvrdit výchozí hodnotu. V našem případě budeme vytvářet oddíl s číslem oddílu 1. Po výběru těchto možností budeme dotázáni na specifikování začátečního sektoru našeho oddílu. Při tomto

---

[15] GPT disk je disk, který používá metodu pro rozdělení disků na oddíly zvanou GUID Partition Table (GPT). Tato metoda nahrazuje tzv. MBR (Master Boot Record). Mezi výhody GPT patří možnost větší velikosti disků a téměř nekonečně mnoho oddílů na disku. [35]

[16] Boot partition, neboli zaváděcí oddíl obsahuje nezbytné soubory pro zavádění operačního systému. Většina moderních počítačů ho využívá k inicializaci operačního systému během startu počítače. Tento oddíl obvykle obsahuje zaváděcí záznam (boot loader), který se spouští při startu počítače a načítá operační systém do paměti.

dotazu máme dáno na výběr z přístupného volného místa a máme zadanou výchozí hodnotu, kterou je první volný sektor v našem systému. Pro výběr výchozí hodnoty stiskneme klávesu **Enter**. V dalším kroku vybereme poslední sektor. Pro výběr zbylého volného místa můžeme stisknout klávesu **Enter**, my si však v tomto návodu vytvoříme oddíl o velikosti 10 MB. Toto provedeme příkazem **+10M.** Pokud bychom nezadali **M** jako MB, bude systém předpokládat, že se tím míní 10 sektorů. Dále stisknutím klávesy **Enter** potvrdíme výchozí hodnotu GUID.

```
kayla@kayla-virtual-machine:~$ sudo gdisk
GPT fdisk (gdisk) version 1.0.8

Type device filename, or press <Enter> to exit: /dev/sda3
Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: not present

Creating new GPT entries in memory.

Command (? for help): n
Partition number (1-128, default 1):
First sector (34-40886238, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-40886238, default = 40886238) or {+-}size{KMGTP}: +10M
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'
```

Obrázek 5: příklad zapsání řešení v terminálu (zdroj: autor)

Po těchto či více krocích, které jsme chtěli provést, je třeba potvrdit a uložit tyto změny. Všechny kroky do tohoto bodu byly uloženy pouze v paměti, ne na disku. Před jejich uložením je vhodné si zkontrolovat, zda v nich nemáme chybu pomocí příkazu **p**. Při uložení nových změn se může stát, že bychom přepsali již existující oddíly, a tím znepřístupnili jejich data.

```
Command (? for help): p
Disk /dev/sda3: 40886272 sectors, 19.5 GiB
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): B1D57124-18B0-465C-8E00-187840B61B17
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 40886238
Partitions will be aligned on 2048-sector boundaries
Total free space is 40865725 sectors (19.5 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048           22527   10.0 MiB    8300  Linux filesystem

Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!

Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/sda3.
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.
```

Obrázek 6: výstup příkazu p a uložení změn (zdroj: autor)

Pro nás je důležité zkontrolovat, zda jsme oddíl správně vytvořili na správném místě, s vyžadovanou velikostí a typem oddílu.

Pokud se vše shoduje s našimi požadavky, můžeme použít příkaz **w**, který uloží a zapíše změny na disk. Pokud bychom se změnami nebyli spokojeni a chtěli je opustit, využijeme příkazu **q**, který opustí příkaz **gdisk** bez uložení změn. Pokud bychom oddíl chtěli odstranit, protože ho již nepotřebujeme a chceme jej i jeho data smazat, použijeme příkaz **d**.

Po vytvoření oddílu je doporučeno jej naformátovat, protože využívání jiného souborového systému může v budoucnu způsobit problémy. Toto uděláme pomocí příkazu **sudo mkfs -t [typ souborového systému] [cesta k disku]**. Mezi možnostmi souborového systémy jsou ext4, xfs nebo Btrfs.

Formát ext4 (Fourth Extended Filesystem) je jeden z nejpoužívanějších typů souborových systémů. Ext4 je považován za výkonný a spolehlivý souborový systém vhodný pro osobní počítače i pro servery.[36] XFS je v některých ohledech lepší než

ext4. Mezi ně patří větší velikost souborů a oddílů a dynamicky alokované inody[17]. Má také některé nevýhody, které ext4 nemá. XFS například nepodporuje journal checksum, který je používán na kontrolu správnosti dat. [38] Btrfs (B-tree Filesystem) je moderní souborový systém, vyvinutý za účelem poskytnutí pokročilých funkcí a vylepšení oproti tradičním souborovým systémům jako ext4. [39]

Výběr typu souborového systému záleží na preferenci uživatele a požadavcích systému. Dále také záleží na operačním systému či systémech, které na zařízení chceme používat, jelikož ne všechny souborové systémy jsou podporovány všemi operačními systémy. Ext4, XFS i Btrfs jsou použitelné na Linux zařízeních. Na Windows a macOS zařízeních nejsou standardně podporované, ale můžeme je pomocí third-party nástrojů používat. [40] Tyto systémy sice nemají všechny funkce jako ext4, XFS či Btrfs, ale pro velkou část uživatelů budou dostačující. [41]

V našem příkladu vybereme ext4 příkazem **sudo mkfs.ext4 [cesta k oddílu]** nebo příkazem **mkfs -t ext4 [cesta k oddílu]**.

Připojení oddílu k souborovému systému

Po naformátování oddílu je také důležité jej připojit k souborovému systému, aby si náš operační systém mohl spojit náš oddíl s jistým adresářem v hierarchii souborového systému. Pokud bychom toto neprovedli, nedokázali bychom se dostat k souborům na daném oddílu. Toto provedeme tak, že využijeme či vytvoříme adresář, na který budeme chtít připojit souborový systém. Před úpravou a uložením nových změn v souboru */etc/fstab* je vhodné uložit kopii souboru, protože pokud do tohoto souboru vložíme nesprávné záznamy, můžou při bootování nastat problémy. V takovém případě můžeme využít tzv. rescue session a změnit nové změny zpátky na naší kopii starého souboru.

Upravíme ho pomocí příkazu **sudo vi /etc/fstab**, kam vypíšeme nový řádek (po stisknutí klávesy **i** na použití módu vložení) ve tvaru

---

[17] Inoda je typ datové struktury na disku, která uchovává informace o souboru nebo adresáři kromě jeho jména a dat. Každý soubor nebo adresář má přiřazenou právě jednu inodu. [37]

<adresa oddílu> <adresa adresáře><typ filesystemu> <možnosti> <dump> <pass> (například: **/dev/sda1/mnt/mynewpartition ext4 defaults 0 2**).

Do kategorie možnosti je nejlepší zvolit *defaults,* pokud není třeba speciální možnosti. Kategorie dump je většinou nastavena na 0 a je využívána nástrojem pro zálohování výpisů. Kategorie pass určuje, v jakém pořadí jsou provedeny kontroly souborových systémů při bootování. Pokud je nastavena hodnota 0, není souborový systém kontrolován. Hodnota 1 je kontrolována jako první[18]. Hodnotu 2[19] systém kontroluje až po zkontrolování všech souborových systémů s hodnotou 1 a dále postupuje obdobným principem. Souborový systém se stejnou hodnotou jsou v dnešní době kontrolovány paralelně.[42]

Po zadání naší úpravy souboru jej uložíme pomocí stisknutí klávesy **Escape**, napíšeme **:wq** a znovu stiskneme **Enter**. Pro otestování správného napojení využijeme příkazu **sudo mount -a**, který se pokusí připojit všechny souborové systémy, jež jsou v souboru /etc/fstab. Pokud výstup hlásí error zprávy, nastala při zadávání úprav chyba a je třeba tyto zatím provedené změny opravit. Pro kontrolu provedeme příkaz **mount | grep [adresa adresáře]**. Výstupem tohoto příkazu by měl být řádek s adresou oddílu, adresou adresáře, na který je napojen, a typem souborového systému. [43]

Swap oddíly a swap soubory

Swap soubory a oddíly se používají pro swapování, což je proces, při kterém operační systém přesouvá data z paměti RAM na pevný disk. U dat, která jsou v danou dobu méně důležitá či méně používaná, dojde k jejich swapování. Swap je využíváno při přetížení RAM či při hibernaci počítače. Při hibernaci počítače se celá RAM uloží do swap místa. Toto pomáhá vypnutí počítače, ale také zachování rychlosti obnovení předešlého stavu. Rozdíly mezi swap soubory a oddíly jsou hlavně v jejich velikosti a možnosti úpravy. Swap oddíl má velikost určenou při jeho vytváření a není možné ji snadno upravit. Oproti tomu swap soubor může mít svoji

[18] Pro root souborový systém je hodnota typicky nastavená na 1, aby byl zkontrolován jako první.
[19] Pro adresáře /mnt nebo /home a další 'non-root' souborové systémy je často nastavena hodnota 2.

velikost kdykoli změněnou a tím je více flexibilní. I přes to jsou swap oddíly častěji používané, a proto se u nich můžou objevit bugy se souborovými systémy. [44]
V dalších kapitolách si ukážeme, jak vytvořit swap oddíly i swap soubory. Výběr mezi těmito variantami je na preferencích uživatele a požadavcích systému.

Vytvoření swap oddílu

Vytvoření swap oddílu a normálního oddílu je téměř totožné, tudíž zde některé kroky nebudou vypsány. Vytvoření swap space začneme tím, že vytvoříme oddíl a přenastavíme jeho typ na 82 (linux swap). Poté ho inicializujeme jako swap oddíl pomocí příkazu **mkswap [adresa oddílu]** a povolíme swapování na swap oddíl příkazem **swapon /swapfile**. Dále je třeba změnit soubor /etc/fstab tak, aby využíval swap oddílu pomocí záznamu **<adresa swap oddílu> swap swap defaults 0 0**. [45]

Vytvoření swap souboru

Pro vytvoření swap souboru využijeme příkazu dd if=/dev/zero of=/var/swapfile bs=1M count=100. V tomto případě vytvoříme soubor o velikosti 100 MB. Poté soubor inicializujeme příkazem **mkswap /swapfile** a povolíme swapování **swapon /swapfile**. Dále je třeba zapsat nový swap soubor do souboru /etc/fstab pomocí záznamu **[adresa swap souboru] swap swap defaults 0 0.** [46]

6.2 *Systémové logování*

Systémové logování sleduje všechny aktivity, které se na našem zařízení provádí a zapisuje je do souborů. Tyto soubory jsou generované systémem nebo aplikacemi, které na něm běží a většinou jsou skladovány v adresáři */var/log*. Soubory také obsahují i status výstupu dané aktivity, čímž nám mohou být nápomocné při hledání chyb. [47] Na zařízeních s operačním systémem Ubuntu je systémové logování zpracováváno syslog démonem.
Jak již bylo řečeno, systém i aplikace mohou generovat své logovací soubory, které lze procházet v případě problémů. Pro základní orientaci v těchto souborech se podíváme na některé logovací soubory běžné pro daný systém. Tyto soubory obsahují informace o autorizacích, systémových démonech a zprávách. Logovací

soubor, který se zabývá autorizačním systémem je */var/log/auth.log*. Můžete v něm například najít všechny pokusy o přihlášení a použití **sudo** příkazu. [48] Dalším souborem je démon log soubor, jenž vypisuje informace o aktivitách démonů[20]. Mezi tyto démony můžou patřit SSH relace, tiskové služby, Bluetooth aj. Pro zobrazení logů z kernelu, slouží soubor */var/log/kern.log*. Posledním důležitým souborem je */var/log/syslog*. Ten obsahuje všechny všeobecné zprávy systému. Lze zde najít informace či data, která nelze najít ve více specializovaných souborech. Dále je pro systémové logování důležitá služba logrotate, jež spravuje rotování a komprimování souborů, které by jinak zaplnily veškeré dostupné místo na disku. Díky této službě si u některých log souborů, které nejsou vlastněny systémem, můžeme nastavit, jak často budou rotovány a kolik jich bude uložených. Defaultně je *longrotate.conf* nastaven na týdenní rotaci log souborů, poslední 4 logy jsou schraňovány a po rotaci souborů se automaticky vytvoří nový prázdný soubor.

6.3 *Plánování úkolů pomocí cron*

Existují úkoly, které je třeba provádět opakovaně či úkoly, které ke svému provedení nepotřebují uživatele. K tomuto účelu využíváme plánování úkolů a cron démona, jenž daný úkol spustí a následně provede. Mezi tyto úkoly patří zálohování, aktualizování či údržba systému, ale také automatické hlášení o stavu systému či webové stránky. Úkol může být téměř jakýkoli.

Pro naplánování nového úkolu využijeme příkazu **crontab -e**, který upraví již stávající crontab soubor. Crontab soubor je tabulka, jež obsahuje všechny plánované úkoly pro daného uživatele a dobu intervalu použití úkolu. Editace tohoto souboru probíhá pomocí vi editoru. V hlavičce souboru je nastavení prostředí, které bude využíváno k provedení úkolu. Mezi nastavení prostředí musíme nastavit proměnnou *shell*, kterou uživatel zamýšlí využít a proměnnou *path*, která označuje, v jakých adresářích může systém hledat skript či příkaz. Po hlavičce již následuje tělo daného souboru, kde je na každém řádku záznam ve tvaru **[minuty] [hodiny] [den v měsíci] [měsíc] [den v týdnu] [příkaz].** Položka den v týdnu začíná

[20] Démon je program, který běží na pozadí a většinou bez interakce s uživatelem. [49]

hodnotou 0 pro neděli. Místo jakékoli hodnoty, kterou není potřeba nastavit či kterou nelze nastavit, necháme znak '*'. Poté existují i speciální výrazy, kterými můžeme definovat intervaly. Mezi tyto patří **@reboot**, **@yearly** nebo **@annually**, **@monthly**, **@weekly**, **@daily** nebo **@midnight** a **@hourly**. [50]

6.4 *Praktická úloha*

V této praktické úloze si ukážeme plánování úloh pomocí cron.

Naším úkolem je vytvořit cron úlohu, která se každou hodinu zeptá uživatele „Dej mi sušenku.“ A uživatel musí napsat „sušenka“ jinak se úloha bude znovu ptát, dokud to uživatel správně nevyplní.

6.4.1 Návod

Nejdříve vytvoříme skript s název **susenka** pomocí příkazu **vi susenka.sh**. Do něj vložíme tento text:

Until [„$input“ == „susenka“]; do

Echo „dej mi susenku“

Read input

Done

Uložíme soubor (nejdříve stiskneme klávesu **Esc** pro opuštění insert módu) pomocí příkazu **:wq.** Dále skript upravíme na spustitelný pomocí **chmod +x susenka.sh**. Dále otevřeme crontab soubor (**crontab -e**) nejspíše pomocí nano a do něj přidáme tento text:

0 * * * * export DISPLAY=:0 && /usr/bin/x-terminal-emulator -e [cesta ke skriptu]. Čas můžeme upravit na danou minutu v které chceme skript spustit, ale tímto Ose bude skript spouštět vždy v celou hodinu. Insert mód opustíme stisknutím kláves **Ctrl** a **O**. Poté stisknutím klávesy **Enter** potvrdíme název souboru a poté stiskneme klávesy **Ctrl** a **X** pro opuštění nano editoru. Poté musíme restartovat cron pomocí **sudo service cron reload** a zadáme naše heslo.

6.5 *Otázky na ověření znalostí*

1 Jak zobrazíme informace o CPU?

a) Pro zobrazení informací o CPU využijeme příkazu **lscpu**, který například vypíše informace o architektuře CPU, počtu vláken na jádře či výrobce a název. Dále si detailnější informace o CPU můžeme zobrazit soubor */proc/cpuinfo.*

2 Co je swap oddíl a swap soubor?

a) Swap soubory a oddíly jsou využívané systémem pro přesouvání dat z paměti RAM na pevný disk v případě přetížení RAM či hibernace počítače. Data, která jsou přesunována nejsou v danou dobu používaná či jsou vyhodnocena jako méně důležitá. Při hibernaci počítače se celá RAM uloží do swap místa.

3 Jaký je rozdíl mezi swap oddíly a swap soubory?

a) Rozdíl mezi swap oddílem a souborem je hlavně v jejich velikosti, ale také v možnosti úpravy. Swap soubor může mít kdykoli změněnou jeho velikost, zatímco swap oddíl má velikost určenou při jeho vytvoření a nedá se snadno upravit.

4 Proč bychom měli rozdělit disk do oddílů?

a) Rozdělení disku do oddílů je výhodné pro organizaci souborů nebo zlepšení výkonu. V případě korupce oddílu rozdělení disku může pomoci s prevencí ztráty dat. Dále je rozdělení disků nutné při bootování vícero operačních systémů na jeden disk, jelikož každý systém typicky vyžaduje svůj vlastní oddíl.

5 K čemu jsou důležité systémové logy?

a) Systémové logy zapisují a sledují veškerou aktivitu prováděnou na zařízení. Jsou důležité při hledání chyb, jelikož obsahují i statusy výstupu aktivit. Systémové logy jsou generované systémem ale i aplikacemi.

6 Jaké je využití cronu?

a) Cron je využíván na plánování úkolů jako například zálohování či aktualizování systému. Tyto úkoly jsou většinou třeba provádět opakovaně.

7 Konfigurace sítě a zabezpečení systému

Cílem této laboratorní úlohy je vysvětlení základů práce se sítěmi jako jejich konfigurace, routování či odstranění problémů, jenž při práci mohou nastat. Dále se kapitola bude věnovat problematice zabezpečení účtů a systému, a také šifrování.

7.1 *Základy konfigurace sítě*

V minulosti bychom pro činnosti v této kapitole využívali nástroje pojmenované net-tools. Toto seskupení nástrojů bylo spojeno za účelem poskytnutí všech funkcí potřebných pro základní síťování, ale vývoj a strategie jejich použití se lišila, tudíž se v dnešní době obvykle používá jiná kolekce nástrojů, pojmenována **iproute2**. Všechny nástroje této kolekce byly vyvíjeny současně, spojuje je tedy syntax a nenastávají problémy při spolupráci. V této úloze si ukážeme, jak s kolekcí **iproute2** pracovat. Tento návod je zpracován za pomoci článku ze stránky DigitalOcean. [51]
Pro zobrazení všech interface, které naše síť obsahuje, využijeme příkazu **ip addr.** Tento příkaz je vlastně **ip addr show** příkaz. Ve výstupu uvidíme základní informace o každém interface jako například jejich název, index a stav, MAC a IP adresu aj. Pro zobrazení informací o specifickém interface tak můžeme využít příkazu **ip link show [název interface]** a pro zobrazení statistiky o komunikaci interface můžeme využít příkazu **ip -s link show [název interface]**.
Dále lze zobrazit routovací tabulku, obsahující informace o směrování datových balíčků v počítačové síti. Toto lze provést příkazem **ip route show**. Ve výstupu najdeme informace jako cílovou síťovou adresu a masku její podsítě, next-hop adresu, interface a metriku[21]. Většinou zde najdeme i tzv. výchozí trasu, jež je poslední variantou trasy. Toto se využívá hlavně při využití přístupu k internetu, jelikož adresy na internetu nebývají uvedeny v routovací tabulce. Pokud bychom chtěli přidat novou trasu do této tabulky, zadáme příkaz **ip route add [cílová adresa či síť] via [next-hop IP adresa] [interface] [id routovací tabulky]**. Interface a ID routovací tabulky jsou volitelné komponenty příkazu. Pokud

[21] Metrika určuje kvalitu dané cesty. Čím je tato hodnota nižší, tím obvykle znamená lepší cestu. Na této pozici také můžeme mít hodnotu váhy, avšak čím je hodnota váhy nižší, tím znamená horší cestu.

nezadáme, přes který interface chceme směrovat pakety, pokusí se je systém směrovat do cílové adresy pomocí výchozí trasy. ID routovací tabulky je třeba zadávat, pokud máme vícero routovacích tabulek a není žádoucí přidat novou trasu do hlavní routovací tabulky. [51]

Dále si zobrazíme soubor **/etc/services** pomocí příkazu **cat /etc/services**. Tento soubor přiřazuje protokoly a názvy známých síťových služeb k číslům portů. Soubor je strukturně jednoduchá tabulka. Typicky má dva sloupce, ale může mít až 4 sloupce s informacemi o názvu služby, čísle portu/protokolu, dále také alias služby a komentář. Název služby je tvořen zkratkou názvu, např. *http* pro webové servery či *ftp* pro file transfer protocol servery. Číslo portu/protokolu je sloupec, který páruje číslo portu s protokolem (TCP nebo UDP) na němž služba běží. Například *http* bývá na portu číslo 80 s použitím protokolu TCP. Alias služby je volitelný sloupec obsahující alternativní název služby jako např. www pro http. Komentáře jsou dalším volitelným sloupcem a mohou obsahovat další informace o službě většinou začínající '#'.

Dalším krokem bude zobrazení. Toto uděláme příkazem **nslookup [název domény]**. Jako příklad můžeme provést příkaz **nslookup google.com**. Dále si vyzkoušíme příkaz **dig [název domény]**. Znovu vyzkoušíme příkaz s adresou **google.com**. Jak se ale tyto dva příkazy liší? **nslookup** nám zobrazí pouze název domény a její IPv4 a IPv6 adresu, kdežto příkaz **dig** nám vypíše dodatkové informace o době čekání na odpověď serveru a čas zadání dotazu. Tento příkaz je také koncipován jako otázka a odpověď, tudíž jsou odpovědi přehledně řazeny ke správným otázkám v případě složitějších dotazů. [52]

V další části se podíváme na soubor **/etc/resolv.conf**, který je konfiguračním souborem klienta DNS[22]. DNS klient (či DNS resolver) využívá tento soubor k překladu doménových jmen na IP adresy. Jeho hlavními prvky jsou *nameserver*, *domain, search a options. Nameserver* určuje IP adresu DNS serveru, na nějž budou směřovány dotazy, těchto IP adres může být pro zálohu uvedeno více. *Domain* je lokální doménové jméno, ne vždy bude nastaveno. *Search* je seznam domén, které

[22] DNS je hierarchický systém doménových jmen. Tento systém překládá názvy domén webových stránek z číselné podoby na doménové jméno, což je název dané webové stránky. [53]

mají být vyhledány, pokud jméno hostitele není plně kvalifikované. Jsou to vlastně zálohy pro hostitele. *Options* mohou být různé možnosti konfigurace DNS klienta, například počet pokusů či časového limitu. Na některých systémech může být obsah tohoto souboru dynamicky generován, a proto může být jeho manuální editování přepsáno. [52]

7.2 *Konfigurace sítě*

V této kapitole si ukážeme, jak změnit nastavení interface a úpravu systémového hostname za pomocí využití kurzu Linux II od Cisco[1].

V prvním kroku si otevřeme konfiguraci ens3 interface pomocí příkazu **ip addr show ens3**. Poté si zobrazíme konfigurační soubor pro síťové rozhraní příkazem **cat /etc/netplan/01-netcfg.yaml.** V dalším kroku tento soubor zkopírujeme do našeho domovského adresáře: cp **/etc/netplan/01-netcfg.yaml.** Originální soubor následně upravíme pomocí vi editoru: **vi /etc/netplan/01-netcfg.yaml**. Zde přejdeme do insert módu pomocí klávesy **i** a zadáme novou statickou konfiguraci ve tvaru **ens3:** /nl **addresses: [10.10.10.2/24]** /nl **gateway4: 10.10.10.1**. Stisknutím klávesy Esc se vrátíme zpět do výchozího módu, kde pomocí kláves **:wq** uložíme změny a opustíme vi po stisknutí klávesy **Enter**. Pro aplikování nových změn budeme muset provést příkaz **netplan apply**. Poté si zkontrolujeme, zda se konfigurace ens3 interface změnila a můžeme vyzkoušet naše routování příkazem **ip route show**.

Dále si ukážeme, jak zobrazit a změnit systémové hostname. Hostname si zobrazíme pomocí příkazu **hostnamectl**[23]. Tento příkaz nám zobrazí informace o našem statickém hostname, které je uloženo v souboru /etc/hostname a dále např. typ zařízení, ID stroje, typ kernelu a operačního systému. Pro změnu našeho hostname znovu využijeme příkazu hostnamectl s argumenty **set-hostname [název nového hostname]** a pro kontrolu můžeme opět provést příkaz **hostnamectl**, tentokrát však bez argumentů. Restartování zařízení nám změní hostname na nový název.

[23] Malé L na konci slova.

7.3 *Radius server*

Radius server (Remote Authentication Dial-In User Service) je AAA[24] protokol, který je používán pro zvýšení zabezpečení účtu. Tento protokol používá služby RADIUS klient a RADIUS server. Klient je používán na autentizaci uživatelů, zatímco přes server lze spravovat databázi s uživateli. Při pokusu o přihlášení uživatele se klient zeptá serveru, zda se uživatel může připojit. Server toto připojení povolí, pokud je uživatel ověřený k připojení a má vyžadovanou autorizaci. [54]

Proces autentizace a autorizace přístupu standardně obsahuje tyto kroky: autentizace pomocí uživatelského jména a hesla. Klient poté pošle žádost o přístup k serveru, která obsahuje sdílený secret. Server si přečte secret a ověří, že pochází z autorizovaného zdroje. Pokud není z autorizovaného zdroje, zprávu zahodí. Poté server zjistí, jakou metodu autentizace má použít, a pokud je povolena, přečte si přihlašovací údaje uživatele. Ty následně porovná s daty ve své databázi, a pokud najde shodu, vyhledá údaje o autorizaci uživatele. Zpátky odešle klientovi zprávu, která obsahuje sdílený secret, které se musí shodovat s tím od klienta. Pokud se secret shodují, přečte si klient hodnotu Filter ID obsaženou ve zprávě od serveru. Filter ID nám říká, do které skupiny uživatelů patří uživatel podle jejich autorizace. Poté je uživatel autentizován a autorizován, a může používat RADIUS klienta. [55]

7.4 *Šifrování pomocí SSH*

V této kapitole se zaměříme na SSH sessions. SSH, neboli secure shell, je síťový protokol, který se používá zejména pro bezpečné připojení na vzdálený server. [56] Funguje tak, že na místním terminálu budou zadány příkazy, které jsou po připojení odeslány na vzdálený server, kde jsou následně provedeny. Dále se zaměříme na nastavení SSH klíčů. Tyto klíče jsou používány pro autentizaci. Uživatel má tajný soukromý klíč a klíč veřejný. Veřejný klíč musí být zkopírován do souboru ~/.ssh/ v domovském adresáři uživatele. V tomto souboru jsou veřejné klíče, se kterými se lze přihlásit k tomuto účtu. SSH server ověří, zda se uživatel snaží připojit se

[24] AAA protokol se zabývá autentizací, autorizací a účtováním. Autentizace se zabývá ověřováním uživatele. Autorizace udává, co může již ověřený uživatel využívat. Účtování sleduje, jak uživatelé využívají síťové služby.

správným soukromým klíčem, jenž odpovídá veřejnému klíči. Toto udělá pomocí šifrování, kdy si server zkontroluje, který klíč má použít. Dále vygeneruje náhodný řetězec a zašifruje ho pomocí daného veřejného klíče. Tento zašifrovaný řetězec může být odšifrován pouze pomocí spojeného soukromého klíče. Při přijetí tohoto řetězce od serveru se klientův počítač pokusí odšifrovat řetězec pomocí soukromého klíče a zkombinuje odšifrovaný řetězec, v ideálním případě se bude jednat o původní řetězec, s ID dané SSH relace, což vygeneruje MD5 hash hodnotu a pošle ji zpátky serveru. Server již měl ID relace i původní náhodný řetězec, tudíž pouze porovná obě MD5 hash hodnoty. Pokud se tyto hodnoty shodují, jedná se o správný soukromý klíč a klient se může připojit.

Pro generování SSH klíčů používáme příkaz **ssh-keygen -t rsa -b 4096**. Argument **-t** určuje typ klíče jako například **rsa** a argument **-b** určuje délku klíče v bitech. Mezi známými typy klíče jsou RSA, ECDSA a ED25519. Obvykle jsou poreferovány RSA klíče a jsou také výchozí hodnotou. Pokud chceme specifikovat název souboru, přidáme argument **-f** následovaný názvem souboru. Dále můžeme přidat komentář ke klíči pomocí argumentu **-C**. Po zadání příkazu budeme tázáni na soubor, kam chceme uložit dané klíče. Pokud nechceme vybrat žádný specifický soubor, pouze stiskneme klávesu **Enter** a tyto klíče se uloží do výchozího schovaného adresáře *.ssh* v našem domovském adresáři. Toto také umožní našemu SSH klientovi tyto klíče automaticky najít. Dále můžeme využít možnosti tzv. přístupové fráze. Této možnosti bychom měli využít, jelikož tím zvýšíme úroveň zabezpečení. Pokud přístupovou frázi nastavíme, bude třeba ji při každém připojení zadat, pokud nevyužijeme možnosti ssh agenta[25]. Poté se nám vygenerují oba klíče. Veřejný klíč bude uložen v souboru [adresa souboru].pub, zatímco ten soukromý bude v souboru [adresa souboru] bez koncovky pub.

Změnu či odstranění přístupové fráze lze provést pomocí příkazu **ssh-keygen -p**, kde bude poté zadán soubor se soukromým klíčem, jehož přístupovou frázi chceme změnit. Nejdříve zadáme stávající frázi, poté novou frázi, a nakonec potvrdíme naši volbu opětovným zadáním.

---

[25] SSH agent je manažer SSH klíčů. Ukládá si klíče a certifikáty v paměti a není třeba proto vypisovat přístupové fráze při každém připojení na server. [57]

Pro připojení k serveru se používá tzv. otisk. Toto pomáhá k autentizaci, ale také se snaží zabraňovat man-in-the-middle útokům[26]. Pro získání otisku vzdáleného serveru použijeme příkaz **ssh-keyscan -p [číslo portu] [název vzdáleného serveru]**, kde není nutné použít číslo portu a argument **-p**. Výstupem tohoto příkazu bude hash hodnota odvozená od veřejného klíče. Pokud se hodnoty otisku a klíče shodují, je připojení k serveru bezpečné.

Po ověření bezpečnosti připojení navážeme spojení se serverem. Pokud se uživatelské jméno na zařízení shoduje se jménem na serveru, můžeme použít příkaz **ssh [IP adresa vzdáleného serveru]**. Pokud by se uživatelské jméno lišilo, využijeme příkazu **ssh [uživatelské jméno@IP adresa vzdáleného serveru].** Při prvním připojení se zobrazí zpráva o autenticitě serveru a jeho otisk, který zkontrolujeme a pokud se otisk shoduje, je možné se připojit. Dále budeme tázáni na tzv. přístupovou frázi našeho soukromého klíče.

7.5 *GPG klíče*

Tato kapitola se zabírá GPG klíči a je čerpaná z oficiálních GnuPG stránek [59]. GnuPG je nástroj pro bezpečnou komunikaci a zabezpečení souborů. Tento nástroj používá systém veřejného a soukromého klíče. GPG klíče a podobné nástroje je vhodné používat kvůli zvýšení bezpečnosti vaší komunikace, zajištění soukromí a pro ověřování identity.

Pro vygenerování páru klíčů využijeme příkazu **gpg --full-generate-key**. Poté si mezi typy klíčů RSA, DSA a ElGamal vybereme, který klíč chceme použít. DSA slouží pouze pro podepsání a ElGamal pro podepsání a šifrování. Při využití možnosti DSA a ElGamal se vygenerují páry klíčů obou typů, kde DSA klíče jsou primárními klíči používanými na podepsání a ElGamal klíče jsou podřízené a slouží na šifrování. Ať vybereme jakoukoliv možnost, vždy můžeme později přidat další podřízené klíče na podepisování a šifrování.

Po vybrání typu klíče musíme také vybrat velikost klíče. Klíče typu DSA mají omezení velikosti klíče na hodnotu mezi 512 a 1024 bity. Klíče typu ElGamal nemají

---

[26] Man-in-the-middle útok je útok, kdy se útočník umístí mezi komunikující strany a sleduje či manipuluje přenos dat mezi těmito stranami. [58]

žádné omezení velikosti. Pokud ale využíváme GnuPG, tak je vyžadováno, aby všechny využívané klíče měli velikost větší než 768 bitů. Pokud bychom tedy generovaly klíče typu DSA a ElGamal jejichž velikost klíče by přesáhla 1024 bitů, tak klíč typu ElGamal bude mít velikost o zadané hodnotě, zatímco klíč DSA bude mít hodnotu 1024 bitů.

Dále je třeba vybrat dobu expirace klíčů. Je možné vybrat klíč, jehož doba expirace neexistuje nebo vybrat expiraci v řádech dní, týdnů, měsíců či let. Většina uživatelů využívá možnosti bez expirace, i když to pro ně není nejlepší varianta a měli by nastavit expiraci hesla například na každý půl rok.

Poté je nutné nastavit uživatelské ID, ve kterém bude systém požadovat celé jméno, komentář a emailovou adresu. Při vytvoření klíče je vytvořeno pouze jedno uživatelské ID, je však možné později přidat uživatelská ID. Po zadání těchto dat dále zadáme naši přístupovou frázi. Je ovšem třeba si dát pozor, jelikož uživatelská ID nemůžou být po vytvoření upravena.

Mezi důležitými kroky po vygenerování a nastavení zabezpečení klíčů je vytvoření tzv. odvolacího certifikátu, který můžeme použít v případě kompromitace či ztráty přístupové fráze. Odvolaný veřejný klíč je stále možné využít na ověření bývalých podpisů, ale nemůže být použit na šifrování budoucích zpráv pro uživatele. Také to neovlivňuje schopnost odšifrování zpráv, které byly uživateli poslány v minulosti, pokud má stále přístup k soukromému klíči. Toto provedeme pomocí příkazu **gpg --output revoke.asc –gen-revoke [specifikátor klíče]**, kde lze jako specifikátor klíče využít ID soukromého klíče nebo část uživatelského ID, který identifikuje uživatelovy klíče. Tento certifikát je vygenerován do souboru revoke.asc. Pokud bychom nevyužili možnosti vygenerování certifikátu do daného souboru (využitím argumentu **--output revoke.asc**), bude výsledek vygenerován do standardního výstupu. Certifikát klíče je krátký, je vhodné si jej tedy vytisknout a uschovat na bezpečném místě.

Pro výměnu klíčů s ostatními uživateli použijeme příkaz **gpg --list-keys**. Ve výstupu bude patrný primární veřejný klíč a podřadný veřejný klíč. Pokud bychom chtěli poslat veřejný klíč jako soubor v příloze, exportujeme jej pomocí příkazu

gpg -- output [název souboru] --export [email uživatele]. Tento klíč je exportovaný v binárním formátu, pro exportování ve formátu ASCII, použijeme příkaz

gpg --armor --export [email uživatele].

Pro importování veřejného klíče existuje příkaz **gpg --import [název souboru]**.

Všechny klíče zobrazíme příkazem **gpg --list-keys**, kde uvidíme veřejné klíče uživatele a klíče, které importujeme. Klíče rozeznáme pomocí emailových adres, které jsou ke každé dvojici klíčů připsané. Po importu klíčů je validujeme, což provedeme dotázáním se vlastníka klíče na jeho otisk. Nejdříve musíme být v editovacím módu, pokud budeme chtít klíč validovat, k čemuž využijeme příkazu **gpg --edit [email vlastníka klíče].** Poté pro zobrazení klíče použijeme příkaz **fpr**. Při zobrazení tohoto klíče provedeme s vlastníkem klíče kontrolu shody otisků mimo editovací mód. Pokud jsme si jisti, že máme stejný klíč jako vlastník, můžeme klíč podepsat a tím ho validovat. Verifikace klíčů je slabina kryptografie veřejných klíčů, a proto bychom měli být velmi opatrní, které klíče podepisujeme a vždy bychom měli zkontrolovat otisk s vlastníkem před podepsáním. Podepsání klíče uskutečníme pomocí příkazu **sign**. Prostřednictvím příkazu **check** zobrazíme již podepsané klíče. Zde uvidíme seznam klíčů podle vlastníků, u každého také informace o podepsání.

Dále můžeme GPG klíče použít na šifrování souborů. Soubor zašifrujeme pomocí veřejného klíče uživatele, pro kterého je daný soubor. Uživatel si následně soubor dešifruje pomocí svého soukromého klíče. Tím se zabrání dešifrování klíče jinou osobou než konkrétním uživatelem. Příkaz **gpg --output [název souboru] --encrypt --recipient [email příjemce] doc** je využíván pro zašifrování souboru**.** Argument **--recipient** lze využít pouze pro jednoho uživatele. Pokud chceme umožnit dešifrování daného souboru více uživatelům včetně sebe sama, je třeba připsat všechny uživatele na list příjemců. Pro dešifrování využijeme příkazu **gpg --output doc --decrypt [název souboru]**, který nás vyzve k zadání přístupové fráze**.** [59]

7.6 *Praktická úloha*

V této praktické úloze si vytvoříme soubor, který poté zašifrujeme pomocí GPG klíče. Kroky postupu:

1. Vytvoříme textový soubor s názvem **tajemstvi**, který naplníme textem např. „tajemství“
2. Vygenerujeme veřejný a privátní GPG klíč
3. Zašifrujeme soubor **tajemstvi** a jako příjemce použijeme svoji e-mailovou adresu

7.6.1 Návod

Soubor vytvoříme pomocí **vi tajemstvi.txt**, do kterého napíšeme text „Tajemství“. Tento soubor poté uložíme a zavřeme. Poté si vytvoříme GPG veřejný a privátní klíč pomocí příkazu **gpg --full-generate-key.** Přes možnosti stačí projít stisknutím klávesy Enter při každé možnosti. Poté zadáme informace o uživateli a přístupovou frázi. Zobrazíme si veřejný klíč pro kontrolu správnosti pomocí **gpg --list-keys**, pro privátní klíč **gpg --list-secret-keys**. Poté zašifrujeme soubor pomocí příkazu **gpg --encrypt --recipient [email příjemce] tajemstvi.txt**. Jako email příjemce zadejte svůj email. Po zadání přístupové fráze bude soubor zašifrován veřejným klíčem příjemce. Po zašifrování bude vytvořen nový soubor **tajemstvi.txt.gpg** v daném adresáři a je to daný zašifrovaný soubor.

7.7 *Otázky na ověření znalostí*

1 Proč bychom měli zabezpečovat systém a účty?

   a) Zabezpečení systému a účtů je důležité pro ochranu citlivých dat, zabránění neautorizovaného přístupu či zmírnění možnosti kybernetického útoku.

2 Jaký je rozdíl mezi **net-tools** a **iproute2**?

   a) Kolekce nástrojů **iproute2** byla vyvíjena současně, a proto ji spojuje syntax a nenastávají problémy při spolupráci jejích nástrojů. Oproti tomu kolekce **net-tools** byla vyvíjena různými autory v různém období a je považována za kolekci pouze proto, že dané nástroje byli většinou využívány pospolu jako nejlepší varianta.

3 Co je radius server a k čemu je využíván?

   a) Radius server je používaný pro zvýšení zabezpečení účtu pomocí autentizace, autorizace a accounting. Autentizace je standardně prováděná pomocí uživatelského jména a hesla. Autorizaci provádí server pomocí předem definovaných pravidel v databázi.

4 Jaký je rozdíl mezi autentizací a autorizací?

   a) Autentizace je proces ověřování uživatele pomocí představení přihlašovacích či přístupových údajů. Autorizace udává, co může ověřený uživatel využívat tedy je kontrolou přístupu.

5 Co je SSH a k čemu je využíváno?

   a) SSH je síťový protokol, který je používán pro bezpečné připojení mezi dvěma zařízeními přes nezabezpečenou síť. SSH může být využíváno například pro vzdálený přístup k zařízení či bezpečný přenos souborů.

6 Jak fungují GPG klíče?

   a) GPG klíče jsou využívané pro bezpečnou komunikaci a zabezpečení souborů. K tomuto účelu je používán systém veřejného a soukromého klíče. Uživatel si vygeneruje pár soukromého a veřejného klíče. Soukromý klíč by měl znát jenom daný uživatel, zatímco veřejný klíč může být sdílen s ostatními uživateli. Veřejný klíč je používán na šifrování zprávy či dat příjemce. Zprávu, která je zašifrována veřejným klíčem může dešifrovat pouze

vlastník odpovídajícího soukromého klíče. Šifrování pomocí soukromého klíče umožňuje podepsání zprávy či dat, a tedy ověření jejich pravosti.

8 Práce se soubory

8.1 *Globování souborů*

Globování souborů je funkce, jež vyhledává soubory podle zadaných vzorů. Mezi vzory může být použit jakýkoli počet znaků, jenž lze zadat. Glob je řetězec obsahující vzor a další řetězec. Mezi globy se může použít například typ souborů, se kterým chceme pracovat. Pro označení vzorů se používají znaky '*****', '**?**', '**[ ]**' a '**!**'. Zástupný znak hvězdička '*****' se používá pro počet znaků 0 až nekonečno, což znamená, že tato pozice může být neobsazena. Znak otazník '**?**' zastupuje přesně jeden znak. Mezi hranatými závorkami je uveden rozsah znaků či vícero rozsahů, ze kterých systém vybere jeden či více znaků. Dále zde může být uveden rozsah nedovolených znaků. Lze zadat čísla i písmena, ovšem je třeba rozlišit malá a velká písmena. Vykřičník vylučuje znaky, které jsou specifikované v hranatých závorkách.

Využití můžeme ukázat na příkladu přesunutí všech .txt souborů z aktuálního adresáře do adresáře text. Toto bychom udělali příkazem **mv *.txt text/**. Pokud bychom chtěli, aby daný .txt soubor neměl zadaný počet znaků a začínal malým písmenem 'a', upravíme příkaz na **mv [a]*.txt text/**. Pokud by ale počáteční znak měl být cokoli kromě písmen 'a' až 'n', tak příkaz bude vypadat **mv [!a-n]*.txt text/**. [60]

8.2 *Regulární výrazy*

Regulární výrazy (regexy) jsou sekvence znaků, které definují vzor vyhledávání[61]. S regulárními výrazy se potkáme ve velké většině programovacích jazyků a jsou využívané převážně na nacházení shody řetězců v komplikovaných podmínkách. Jednotlivé programovací jazyky mohou používat mírně odlišné vzory. Regexy využíváme například pro ověření řetězců či pro manipulaci s řetězci. Pokud není třeba regexy použít, je lepší se jim vyhnout. [61]

Rozdíl mezi regulárními výrazy a globováním je, že regulární výrazy využíváme pro shody řetězců uvnitř souboru, zatímco globování pro shody řetězců v názvu souboru či jeho typu v terminálu. [62]

Příkladem použití regexu může být příkaz **cat priklad|grep „a\+t“**, kde regex je **„a\+t“** a vyhledává slova, která obsahují znak **a** před znakem **t** ve slově v souboru **priklad**.

8.3 *Práce se soubory a adresáři*

V této kapitole si ukážeme základy práce se soubory a adresáři.

8.3.1 Vypsání souborů

Prvním příkazem je **ls**, který nám vypíše všechny soubory v aktuálním adresáři s výjimkou skrytých souborů. Pokud bychom chtěli mít na výstupu pouze soubory nějakého typu či nějakého jména, můžeme příkaz **ls** spojit s globováním. Kromě tohoto lze také využít argumentů příkazu **ls**. Pro zobrazení všech souborů (i těch skrytých) použijeme příkaz **ls -a**, kde budou skryté soubory označené tečkou na začátku. Skryté soubory mohou být konfiguračními soubory, soubory pro verzování či soubory se zálohami, a proto je vhodné si je zobrazit při problémech. Pro zobrazení posledního upravovaného souboru ve výstupu na prvním místě využijeme příkazu **ls -lt**. Abychom zobrazili soubory od nejstaršího data úpravy po nejnovější, přidáme k příkazu **ls-lt** další argument **-r(**reverse), tedy **ls -lrt**. Využitím příkazu **ls-l** si zobrazíme všechny informace o všech souborech v aktuálním adresáři. Toto se nám může hodit pro zobrazení přístupových práv souboru nebo pro zobrazení vlastníka a skupiny souboru. Syntax výstupu je **[typ souboru, přístupová práva] [počet linků] [vlastník] [skupina] [velikost v bytech] [poslední úprava] [název souboru]** První znak výstupu na každém řádku nám také označuje typ souboru, mezi kterými může být normální soubor ‘**-**‘, adresář ‘**d**‘, socketový soubor ‘**s**‘ nebo linkový soubor ‘**l**‘. Přístupová práva jsou spojena s typem souboru ve výstupu a jsou vypsána v absolutním stylu (použití r,w a x), tzv. na výstupu může být například **drw-r-x-r--**. Pro zobrazení velikosti souboru v jednotkách kB, MB či GB využijeme příkazu **ls -lh**. [63]

8.3.2 Určení typu souboru

Určení typu souboru může mít vícero důvodů. Lze jej využít pro detekování škodlivých souborů, pro správné zpracování souboru nebo pro pomoc při

diagnostice problémů spojených s kompatibilitou souborů. Pokud bychom chtěli určit typ souboru, využijeme příkazu **file [možnosti] [název souboru]**. Ve výstupu může být kromě typu souboru vypsaná i verze daného typu souboru nebo další obecné informace. Pro zobrazení pouze typu souboru bez dalších doplňujících informací přidáme argument **-b** (**file -b [název souboru]**). Dále si můžeme zobrazit typy všech souborů v adresáři. Pro aktuální adresář využijeme příkazu **file *** a pro jiné adresáře **[název adresáře]/***. Zobrazení komprimovaných souborů provedeme argumentem **-z** (**file -z [jméno souboru]**) a ve výstupu budeme mít typ souboru. Pokud je daný soubor adresář, bude výstupem komprimovaný archiv, ale tento příkaz nám nezobrazí typy souborů, které jsou obsaženy v daném adresáři. [64]

8.3.3 Příkaz touch

Příkazem **touch** se dá vytvořit nový prázdný soubor nebo změnit informace o času posledního otevření a času úprav souboru. Nejdříve se zaměříme na prázdné soubory. Pro vytvoření jednoho či více prázdných souborů použijeme příkaz **touch,** a poté zadáme požadovaný počet souborů podle jejich jmen, tedy **touch [název prvního souboru] [název druhého souboru]** atd. Všechny vypsané soubory se vytvoří ve stejný čas. Pro vytvoření prázdného souboru také můžeme využít příkaz **cat > [název souboru i s jeho příponou]** nebo příkaz **cat > [název souboru]**. [65] Dále můžeme měnit informace o času. Toto se využívá pro zachování časové značky originálního souboru při kopírování nebo při práci s automatizovanými skripty, kdy můžeme čas změnit tak, aby se skript spustil či nespustil. Časové značky souboru jsou tři, a to čas posledního přístupu, poslední úpravy a poslední změny. Pro změnu času přístupu se využívá argumentu **-a**. Pokud chceme, aby čas posledního přístupu byl aktuálním časem, zadáme příkaz **touch -a [název souboru]**. Čas posledního přístupu však můžeme změnit na jakýkoli jiný čas pomocí příkazu **touch -a -t YYYYMMDDhhmm.SS [název souboru]**. Pro změnu času poslední úpravy využijeme příkazu **touch -m [název souboru],** respektive **touch -m -t YYYYMMDDhhmm.SS [název souboru]** pro změnu na zadaný čas. Čas poslední změny a úpravy jsou dvě různé hodnoty, jelikož čas poslední úpravy udává dobu změny obsahu souboru, zatímco čas poslední změny udává dobu poslední změny

metadat souboru. Metadata souboru mohou kromě časů úpravy či přístupu být také vlastnictví souboru či přístupová práva. Čas poslední změny se automaticky aktualizuje při změnách časů přístupu a úpravy, ale také pokud změníme jakákoli metadata. Tuto časovou značku ale nemůžeme aktualizovat pomocí příkazu **touch**. Pokud bychom chtěli zkopírovat časy z jiného souboru, využijeme příkazu **touch -r [název souboru, ze kterého chceme data kopírovat] [název souboru, kam chceme data zkopírovat]**. [65]

8.4 *Operace se soubory*

V této kapitole se zaměříme na kopírování, přesun a odstranění souborů, a také na hledání souborů.

8.4.1 Hledání souborů

Pokud hledáme soubor, adresář či jakýkoli objekt na našem zařízení, je vhodné použít příkazu **find,** se kterým můžeme procházet soubory podle jména, typu, velikosti, času poslední úpravy, vlastnictví či kombinací těchto kritérií. Můžeme také využít příkazu **locate**, který je ve hledání rychlejší a přesnější. Příkaz **find** prohledává celý systém, zatímco příkaz **locate** prohledává databázi s cestami souborů a nekontroluje, zda soubor stále existuje či zda existují nově vytvořené soubory, které prozatím nejsou v databázi. Oba příkazy tedy mají své využití, a proto si vysvětlíme, jak je použít.

Příkaz find

Hledání souboru podle jména lze provést pomocí příkazu **find [název adresáře] -name [název souboru]**, kdy je hledání citlivé na velká a malá písmena nebo pomocí **find [název adresáře] -iname [název souboru]**, kdy hledání není citlivé na velikost písmen. Pokud neznáme plný název, můžeme také využít globování.

Hledání podle typu objektu provádíme pomocí argumentu **-type** a jeho parametrů (**find [název adresáře] -type [parametr]**. Mezi jeho používané parametry patří **f** jako normální soubory, **d** jako adresáře nebo **l** jako symbolické linky.

Pro vyhledávání podle velikosti používáme příkaz **find [název adresáře] -size [velikost]**, kde zadáme hodnotu velikosti souboru a před ni přidáme znaménko plus nebo mínus pro soubory větší či menší než naše hodnota velikosti. Pokud bychom chtěli, aby velikost byla v určitém rozsahu, můžeme využít argumentu **-size** dvakrát ve stejném příkazu, a to například **find [název adresáře] -size +40M -size -50M**. Jednotky velikosti zadáváme jako **c** pro byte, **k** pro kB, **M** pro MB a **G** pro GB.

Vyhledávání podle času poslední úpravy se provádí příkazem **find [název adresáře] -mtime [počet dní]**, kde počet dní s použitím znaménka mínus znamená před nejvíce kolika dny mohl být soubor upraven. Pokud bychom použili znaménko

plus místo mínus, vyhledávání nám zobrazí soubory, které byly upraveny před více dny, než jakou hodnotu jsme zadali.
Vyhledávání souborů podle vlastnictví může být užitečné, pokud bychom daného uživatele potřebovali smazat a chceme jeho soubory převést na jiného vlastníka. Soubory vyhledáme pomocí příkazu **find [název adresáře] -user [uživatelské jméno]**.

Příkaz locate

Hledání souboru podle jména také můžeme provést pomocí příkazu **locate [možnosti] [vzor/název souboru]**. Pokud příkaz najde shodu, status výstupu bude 0, pokud však příkaz nenajde schodu nebo při provádění proběhne chyba, tak status výstupu bude 1. Příkaz můžeme provést s celým názvem souboru i jeho příponou nebo můžeme využít vzorů a globování. Dále můžeme nastavit citlivost na malá a velká písmena pomocí argumentu **-i**. [66]
Pro zlepšení našeho vyhledávání můžeme aktualizovat databázi před vyhledáváním, což provedeme příkazem **sudo updatedb**. [67]
Pokud bychom věděli, že výstup příkazu bude mít vícero výsledků, můžeme limitovat počet výsledků. Pro to využijeme příkazu **locate [název souboru/vzor] | head -n [počet]**. Pro vyloučení některých adresářů z databáze, a tedy jejich vyloučení z prohledávání, upravíme konfigurační soubor */etc/updatedb.conf*. [68]

8.4.2 Kopírování souborů

Tato část bude zaměřená na kopírování souborů pomocí příkazu **cp**. Syntax tohoto příkazu je **cp [název souboru s příponou] [cílová lokace]**. Pro zkopírování vícero souborů tento příkaz také můžeme použít ve verzi **cp [název prvního souboru s příponou] [název druhého souboru s příponou] ... [název posledního souboru s příponou] [cílová lokace]**. Příkaz **cp** rovněž můžeme použít na kopírování adresářů i s jejich obsahem. Příkaz musíme upravit na **cp -r [název adresáře] [cílová lokace]**, kde argument **-r** musí být použit pro rekurzivní kopírování souborů z adresáře. Při kopírování také můžeme využít globování souborů a všechny soubory, které splňují daný vzor zkopírovat do našeho cílového adresáře. [69]

8.4.3 Přesouvání souborů

Dalším typem manipulace souborů je jejich přesouvání. Mezi přesouvání se také počítá přejmenování souboru, jelikož jej tzv. přesuneme ze souboru s původním jménem do souboru s jiným jménem, což provedeme pomocí příkazu **mv [možnosti] [název souboru] [cílová lokace]** případně **mv [možnosti] [název prvního souboru]...[název posledního souboru] [cílová lokace]** pro přesunutí vícero souborů. Příkazem **mv** také můžeme přesunout adresář do jiného adresáře (**mv [možnosti] [název adresáře] [cílová lokace]**). Pro přesun adresářů je potřeba příkaz použít s argumentem **-r** pro rekurzivní přesun všech jeho souborů a podsložek. Pro přejmenování souboru můžeme využít příkazu **mv [název souboru] [nový název souboru]** a dokonce ho můžeme přejmenovat a přesunout zároveň příkazem **mv [název souboru] [název adresáře/nový název souboru].** Tímto příkazem také můžeme přejmenovat adresář (**mv [název adresáře] [nový název adresáře]**). [70]

8.4.4 Odstranění souborů

Pro odstranění souborů, adresářů a dalších objektů používáme příkazu **rm**. Příkazem můžeme smazat jeden či vícero souborů - **rm [název prvního souboru s příponou] ... [název posledního souboru s příponou]**. Tento příkaz pracuje potichu, tzv. nezeptá se nás na potvrzení smazání zadaných souborů. Pokud bychom chtěli, aby se toto potvrzení před smazáním zobrazilo, musíme využít argumentu **-i** (**rm -i [název souboru]**). V tomto případě stisknutím klávesy Y potvrdíme smazání, stisknutí jakékoli jiné klávesy soubor nesmaže. [71]

8.4.5 Práce s adresáři

Kromě příkazů pro práci s adresáři již představených v předchozích kapitolách také existují příkazy **mkdir** a **rmdir**, které jsou speciálně vytvořené pro práci s adresáři. První příkaz **mkdir** je používán pro vytváření nových adresářů a nastavení jejich přístupových práv. Syntax základního příkazu na vytvoření nového adresáře je **mkdir [možnosti] [název adresáře]**, což vytvoří adresář do našeho aktuálního adresáře. Pokud bychom chtěli mít adresář s určitými rodičovskými adresáři, které můžou a nemusí již existovat, využijeme argumentu **-p**, který neexistující adresáře z

příkazu vytvoří. Pokud bychom chtěli mít adresářovou strukturu prvni/druhy/treti, použijeme příkaz **mkdir -p prvni/druhy/treti**. Pro nastavení přístupových práv při vytváření adresáře využijeme argumentu **-m** za použití absolutního nebo symbolického formátu (**mkdir -m [přístupová práva] [název adresáře]**).[72]
Druhým příkazem je **rmdir**, jenž používáme pro odstranění adresářů. Tento příkaz má syntax **rmdir [název adresáře]**, kde po použití příkazu může vyskočit chybové hlášení, pokud není daný adresář prázdný, a takový adresář nebude smazán. Pokud bychom chtěli smazat adresář, který není prázdný, nemůžeme využít příkaz **rmdir**, ale musíme využít již zmíněného příkazu **rm -r [název adresáře]**. [73]
Pro smazání adresáře také můžeme využít příkazu **rm**, který musíme použít s rekurzivní funkcí mazání souborů, tj. příkaz **rm -r ***, pro smazání aktuálního adresáře nebo **rm -r [plná cesta k adresáři]** pro smazání jakéhokoli jiného adresáře. [71]

8.5 *Komprese souborů*

Komprese souborů nebo adresářů se používá pro zmenšení jejich velikosti, a tím pádem i snadnějšímu a rychlejšímu sdílení těchto souborů nebo z důvodu šetření volného místa na disku. Některé formáty souborů, zvláště mediální soubory jako JPEG nebo MP4, již komprimované jsou, tudíž jejich komprese není potřebná a někdy i může zhoršit jejich kvalitu.
Pro kompresi souborů se používají příkazy **tar**, **zip**, **gzip**, **bzip2** a **xz,** z nichž vybíráme správnou možnost pro naši kompresi podle typu komprese, kterou chceme využít. V této kapitole si vysvětlíme rozdíly mezi těmito různými kompresemi a kdy je vhodné je využít. [74]

8.5.1 Typy komprimovaných souborů

Mezi nejznámější typy kompresních nástrojů patří **zip**, **gzip**, **bzip2** a **xz**. Rozdíl mezi nimi je jejich využití, jejich kompresní algoritmus a kompresní poměr. Přestože to není kompresní nástroj, můžeme sem přidat i příkaz **tar**, jelikož jej můžeme spojit s jinými kompresními nástroji. Většinou se **tar** používá s gzip, bzip2 nebo xz.
Asi nejznámější a nejpoužívanější kompresní nástroj je **zip**. Mezi jeho přední vlastnosti kromě přijatelného kompresního poměru a jeho rychlosti patří také jeho

podpora v různých operačních systémech. [75] Gzip a bzip2 jsou typy kompresních nástrojů, které jsou podobně schopné. Gzip je oproti bzip2 rychlejší při kompresi a dekompresi, ale bzip2 oproti tomu poskytuje lepší kompresní poměr. Gzip také podporuje rekurzivní kompresi27. [76] Kompresní nástroj xz poskytuje nejlepší kompresní poměr ze zmíněných typů, ale je výrazně pomalejší. Další jeho nevýhodou je, že není standardně nainstalován ve všech Linux distribucích a je třeba jej instalovat později. [77]

Výběr správného kompresního nástroje záleží na dané situaci a na preferenci uživatele. Pokud nemá uživatel danou preferenci a chce, aby s daným komprimovaným souborem nebyl problém na jiných platformách, je **zip** soubor nejlepší možnost.

Dále by bylo vhodné zmínit konvenci koncovek souborů. Toto je využitelné při dekompresi souborů, aby byl soubor dekomprimován správným typem komprese. Pokud je daný soubor archivním souborem, bude mít koncovku **.tar**. Soubory komprimované zip mají koncovku **.zip**. Pro bzip2 je koncovka **.bz2** a pro xz je koncovka **.xz**. Pokud je soubor archivovaný a zároveň komprimovaný, budou koncovky v tomto pořadí **.tar.[typ komprimace].**

8.5.2 Archivní soubory

Archivní soubory se používají na seskupení vícero souborů a adresářů do jednoho souboru pro zlepšení organizace souborů. Tyto soubory také mohou být komprimované a poté i mezi jejich výhody patří zmenšení velikosti tohoto souboru oproti jejich plné velikosti. Klasické archivní soubory s příponou .tar ale nejsou komprimované. Soubory, které jsou zarchivované mají zachovaná svá metadata. Jak je zřejmé, oba typy těchto souborů jsou spolu provázané v použití. Komprimované soubory se používají na zmenšení velikosti kompresivními algoritmy. Archivní soubory se používají zejména na balení vícero souborů do jednoho.

---

[27] Rekurzivní komprese provádí kompresi všech souborů a podadresářů daného adresáře. To znamená, že bude zachovaná struktura adresářů a zároveň budou všechny soubory zkomprimované.

8.5.3 Tar příkaz

Tar příkaz se využívá na archivaci souborů, a také tímto příkazem můžeme archivní soubory i zkomprimovat, dekomprimovat, udržovat a upravovat. Důrazně se doporučuje používat argument **-f** pro specifikování názvu archivního souboru a **-c** pro vytvoření nového archivního souboru. Pro vytvoření nového nekomprimovaného archivního souboru použijeme tar příkaz ve tvaru **tar -cf [název archivního souboru] [název souborů nebo adresářů pro zarchivování]**. Dekomprimace archivního souboru se provede příkazem **tar -xf [název archivního souboru],** kde argument **-x** extrahuje soubory z archivu. [74]

Díky tomu je zřejmé, že je třeba nejdřív soubor dekomprimovat daným typem komprese a poté dekomprimovat tar příkazem pro archivní soubory.

Pro využití komprese při archivování souborů použijeme argument pro daný typ komprese. **-z** je pro gzip kompresi, **-j** je pro bzip2 a **-J** pro xz kompresi. Zip komprese nemůžeme vytvořit pomocí příkazu **tar**. Pro archivaci a kompresi souboru pomocí gzip využijeme příkazu **tar -czf [název archivního souboru] [název souborů nebo adresářů pro zarchivování]** a pro extrahování tohoto souboru použijeme **tar -xzf [název archivního souboru]**. Extrahování můžeme provést i pro vícero souborů pomocí příkazu **tar -xzf [název prvního archivního souboru]...[název posledního archivního souboru]**. Ostatní typy kompresí se používají stejně jen se záměnou argumentů pro kompresi. Další funkce, pro kterou můžeme využít příkazu **tar,** patří vypsání obsahu archivovaného souboru. Toto provedeme zadáním příkazu **tar -tf [název archivního souboru]**. Pokud je daný soubor i komprimovaný, musíme mezi argumenty přiřadit znak pro danou kompresi. [74]

8.5.4 Příkazy pro různé typy kompresí

Pro použití zip komprese využijeme příkazu **zip [možnosti] [název zip souboru] [název souborů nebo adresářů pro komprimování]**. Dále můžeme při vytváření zip souboru využít ochrany obsahu tohoto souboru pomocí hesla, což provedeme příkazem **zip -ex [název zip souboru] [název zip souboru] [název souborů nebo adresářů pro komprimování]** a po zadání tohoto příkazu budeme dotázáni na heslo. Toto šifrování není vhodné pro velmi citlivá data, jelikož je poměrně slabé. Pro dekompresi využijeme příkazu **unzip [název zip souboru]**. [74]

Gzip kompresi souborů provedeme příkazem **gzip [možnosti] [název souborů nebo adresářů pro komprimování]**. K tomuto příkazu nemusíme zadávat název komprimovaného souboru, jelikož defaultně je nastaven název na název originálního souboru s koncovkou **gz**. Standardně jsou originální soubory při kompresi nahrazeny jejich komprimovanou verzí. Pokud bychom originální soubory chtěli zachovat, využijeme argumentu **-c,** tj. **gzip -c [název souboru pro kompresi] > [název komprimovaného souboru]**. Dekomprese bude provedena pomocí příkazu **gzip -d [název komprimovaného souboru]** a dekomprimuje komprimovaný soubor do originální formy. Nebo můžeme pro dekompresi také využít příkazu **gunzip [možnosti] [název souboru]**. [78] Pokud bychom chtěli zachovat komprimovaný soubor při dekompresi, využijeme příkazu **gzip -dk [název komprimovaného souboru]**. [74]

Bzip2 komprese je prováděna pomocí příkazu **bzip2 [možnosti] [název komprimovaného souboru]**. Argumenty tohoto příkazu jsou stejné jako u příkazu gzip tedy **c**, **d** a **k**, ovšem existují i další. Totéž platí pro xz kompresi pomocí příkazu **xz [možnosti] [název souborů nebo adresářů pro kompresi]**. [74]

8.6 *Praktická úloha*

V této praktické úloze si ukážeme prohledávání souborů známé jako zmocnění se vlajky neboli “capture the flag“. Nejdříve najdeme soubor s příponou **.txt** a obsahujícím písmena **o** a **k** v názvu. Vyhledávaný soubor by měl obsahovat větu v českém jazyce. Tuto větu poté zapíšeme do Olivy.

Kroky postupu:

1. Vyhledání souboru s příponou **.txt** a písmeno **o** a **k** v názvu
2. Prohledání vyhledaných souborů
    a. Hledaný soubor obsahuje větu v českém jazyce

8.6.1 Návod

Pro vyhledání souboru s příponou **.txt** a písmeny **o** a **k** v názvu použijeme příkaz **locate -r '\b[oO].*[kK].*\.txt$‘**. Pokud tento příkaz nalezne vícero výsledků, prohledáme manuálně dané soubory. Ve vyhledávaném souboru by měla být jedna věta v českém jazyce. Tímto by se měly eliminovat všechny odlišné výsledky. Danou větou je “Zmocnil jsi se vlajky.“

8.7 *Otázky na ověření znalostí*

1. Jaký je rozdíl mezi globováním a využíváním regulárních výrazů?
    a) Globování souborů vyhledává soubory podle zadaných vzorů nebo-li globů. Regulární výrazy jsou sekvence znaků definující vzor vyhledávání. Regexy jsou více flexibilní a mají komplexnější syntax. Rozdíl je v jejich použití regexy jsou používané uvnitř souborů a zpracování textu, zatímco globování je používáno pro operace správy souborů a skriptování shellu.
2. Na co je využíván příkaz **touch**?
    a) Příkaz **touch** je využíván pro vytváření prázdných souborů a aktualizování informací o čase úprav a posledního otevření souboru.
3. Na co se využívá příkaz **find** a na co příkaz **locate**? Jaký je mezi nimi rozdíl?
    a) Příkazy **find** a **locate** jsou využívané na vyhledání souborů. Rozdíl mezi nimi je v prohledávání systému. Příkaz **find** prohledává celý systém, zatímco příkaz **locate** prohledává již předvytvořenou databázi cest souborů a názvů a to může vytvořit problém s neaktuálními daty. Pro rychlé nalezení souboru je výhodnější využít příkazu **locate**, ale pro komplexnější vyhledávání s pokročilým filtrováním je lepší příkaz **find**.
4. Používá se rozdílný příkaz na přesunování souborů a adresářů?
    a) Pro přesunování souborů je používán příkaz **mv** a na přesun adresářů je využíván příkaz **mv -r**, který musí být použit s argumentem **-r** pro rekurzivní přesun všech souborů a podsložek v adresáři. Proto bychom řekli, že příkaz se používá stejný jenom při přesunu adresářů musí mít příkaz ještě atribut **-r**
5. Jaké typy komprese souborů byly využity v laboratorní úloze?
    a) Mezi využitými typy komprese jsou **zip**, **gzip**, **bzip2** a **xz**.
6. Jaký je rozdíl mezi kompresí a archivací?
    a) Komprese se používá pro zmenšení velikosti souborů či adresářů a jejich rychlejšímu sdílení. Archivace se používá pro organizaci a seskupení souborů a adresářů do jednoho souboru.

9 Administrace systému

9.1 *Uživatelské účty a skupiny*

Pro systémy, které využívá vícero uživatelů s různými přístupy k souborům a adresářům, je vhodné znát principy pro práci s uživatelskými účty a skupinami. Toto je důležité pro bezpečnost, také aby neoprávnění uživatelé neměli přístup k souborům, ke kterým nemají, a neměli tedy možnost tato přístupová práva měnit. Dále také aby nemohli smazat či zobrazit citlivé údaje a pouštět skripty. Rozdělení uživatelů do skupin podle jejich pozic a souborů, ke kterým mají přístup je tedy velmi užitečné.

Uživatelské účty jsou v tomto případě vytvořeny pro jednu osobu, která má své přihlašovací jméno a heslo s přiřazeným uživatelským ID (UID). UID se využívá k identifikaci a přiřazení práv uživatele. K těmto účtům by měla mít přístup pouze daná osoba, pokud se nejedná o uživatelský účet vytvořený pro jeden účel, který vykonává vícero osob, například účet používaný k testování nového systému či servisování stroje. Správně by se to v praxi z důvodu snížení bezpečnosti nemělo provádět, ale běžně se to kvůli usnadnění práce praktikuje. Z důvodu bezpečnosti není také doporučováno, aby vícero lidí používalo stejný účet. V Linuxu jsou dva typy uživatelů. Root uživatel je na systému jen jeden a má všechna privilegia. V některých systémech se již nepoužívá, je nahrazen klasickými uživateli a sudo privilegii.

Běžní uživatelé mají limitovaný přístup k souborům a operacím, které mohou provést. Někteří z normálních uživatelů, také mají práva k použití práv root uživatele pomocí **sudo** příkazu.

Uživatelé také musí patřit do skupin, které rozřazujeme podle práv a privilegií. Každá skupina má své skupinové ID (GID). Skupiny usnadňují práci s přístupy k souborům. [79]

V systému Linux jsou dva typy skupin, a to primární a sekundární. Primární skupiny jsou skupiny, do kterých operační systém přiřazuje nově vytvořené soubory uživatele. Jeden uživatel může mít pouze jednu primární skupinu a musí alespoň do

jedné patřit. Sekundární skupiny jsou dalšími skupinami kam uživatel může a nemusí patřit. Těchto skupin může být až 15. [80]

9.1.1 Vytváření a úprava uživatelského účtu

Uživatelské účty vytváříme při přidání nového člověka do našeho systému či při novém nainstalování operačního systému. Při instalaci operačního systému bychom měli vytvořit root uživatele a neměl by to být náš hlavní účet. Root účet se nedoporučuje jako hlavní, protože má všechny práva a tím pádem může uživatel využívat všech operací a použít něco s čím neví, jak pracovat či jaké jsou důsledky použití. Dále se tím systém otevírá bezpečnostním rizikům. Pokud by útočník získal přístup k našemu účtu, má všechna práva a může zničit celý systém či využít dat, které v něm najde. Mezi bezpečnostní koncepty používané při uživatelských účtech také patří princip nejnižších privilegií, který říká, že by uživatel měl mít nejnižší možná práva pro jeho práci. [81] Tímto se zamezí lidským chybám, a také jsou ochráněna důležitá či citlivá data. Root účet se vytvoří vždy, tudíž je vhodné vytvořit i běžný uživatelský účet, který bude náš hlavní. Nedoporučuje se využívat root účet pro každodenní používání.

Pro vytvoření nového uživatelského účtu využíváme příkazu **sudo useradd [možnosti] [uživatelské jméno].** Možnosti mohou specifikovat skupiny, do kterých tohoto uživatele zařadíme, či datum expirace tohoto účtu. Pro úpravu již vytvořeného účtu můžeme využít příkazu **sudo usermod [možnosti] [uživatelské jméno]**. Většina možností, které lze využít, je stejná jako u předchozího příkazu. Například lze změnit domovský adresář uživatele pomocí příkazu **sudo usermod -d /home/[uživatelské jméno]**. [79]

Pro smazání účtu využíváme příkazu **sudo userdel [možnosti] [uživatelské jméno]**. Mezi používanější možnosti tohoto příkazu patří **-f** nebo **-r**. Možnost **-f** vynutí smazání uživatelského účtu v daný okamžik a odebere uživatelův domovský adresář i tzv. mail spool[28]. I kdyby v danou chvíli byl uživatel přihlášen, můžeme tento účet smazat. Hlavním cílem možnosti **-r** je odstranění domovského adresáře

[28] Mail spool je adresář či systém adresářů, ve kterém dočasně najdeme všechny informace o emailech pro daného uživatele. Sendmail používá mail spooly zatímco qmail ne. [82]

uživatele, všech jeho souborů a také odstranění daného uživatele. Rozdíl mezi těmito možnostmi je v tom, že možnost **-r** dává důraz na smazání souborů, zatímco možnost **-f** dává důraz na odstranění uživatelského účtu a také tím lze obejít některé kontroly, které nám mohou zabraňovat v odstranění daného uživatele. [83]

9.1.2 Správa hesel

Každý uživatel by měl mít na svém účtu nastavené bezpečné heslo, které nikde jinde nepoužívá. Toto heslo by se mělo jednou za čas měnit. V Linuxu pro správu hesel používáme příkaz **passwd**. Nejčastěji je využíván na změnu hesla, ale dá se také použít na úpravu doby platnosti hesla či odblokování zablokovaných účtů. Syntax tohoto příkazu je jednoduché **passwd [možnosti] [uživatelské jméno]**, kde se za možnosti dosadí argument pro danou úpravu hesla a na místě uživatelské jméno bude uživatelské jméno účtu, u kterého chceme změnu provést. Uživatel může upravovat hesla pro jiné účty pouze pokud k tomu má pravomoci. [84]

Změna hesla

Jak již bylo uvedeno výše, změnu hesla pro naše zařízení vykonáme pomocí příkazu **passwd**. Pokud nepoužíváme root účet, budeme muset po zadání tohoto příkazu zadat naše stávající heslo. Poté již můžeme zadat heslo, které chceme nově používat, a ještě ho pro kontrolu zadáme znovu. Pokud jsme všechny kroky zadali správně, vypíše nám terminál zprávu, že změna hesla proběhla úspěšně. Je také dobré vědět, že při zadávání hesla do terminálu nejsou z důvodu bezpečnosti vidět žádné znaky. Pokud bychom měnili heslo pro jiný účet než ten, který zrovna používáme, musíme mít buď **sudo** privilegia nebo musíme používat root účet. Pro změnu hesla jiného uživatele znovu využijeme příkazu (**sudo**) **passwd [uživatelské jméno]**, kde za uživatelské jméno patří jméno účtu, které chceme změnit. Při využití **sudo** privilegií budeme dotázáni na heslo k našemu účtu, aby bylo zabráněno využití tohoto příkazu neoprávněným uživatelem. Tímto příkazem (**sudo passwd root**) také můžeme změnit heslo pro root účet. Pokud bychom chtěli vynutit změnu hesla u některého uživatele, necháme jeho heslo expirovat s okamžitou platností. Uživatel, pro kterého toto nastavíme, si při dalším přihlášení bude muset změnit heslo. Toto vynucení se provede pomocí příkazu **sudo passwd -e [uživatelské jméno]**. Znovu budeme

dotázáni na naše heslo, jelikož využíváme **sudo** privilegií. Pokud bychom chtěli smazat heslo našeho účtu či jiného účtu, využijeme příkazu **passwd -d,** respektive **sudo passwd -d [uživatelské jméno]**. [84]

Uzamčení a odemčení účtu

Pokud bychom z jakéhokoli důvodu chtěli uživateli zamezit přístup k účtu, můžeme jeho účet uzamknout. Mezi důvody uzamčení účtu mohou patřit různá bezpečnostní rizika či překročení pokusů o přihlášení. Pro uzamčení účtu využijeme příkazu **sudo passwd -l**[29] **[uživatelské jméno]**. Pokud by důvody uzamčení účtu již pominuly a chtěli bychom účet znovu zpřístupnit, můžeme jej odemknout pomocí příkazu **sudo passwd -u [uživatelské jméno]**. [84]

Nastavení doby vypršení platnosti hesla

Mezi dalšími používanými argumenty příkazu **passwd** patří **-x**, jenž nastavuje dobu vypršení platnosti hesla v řádu dní. Toto se doporučuje využívat kvůli nebezpečí odhalení hesla či nebezpečí hrozící používáním hesla na vícero účtech. [84]

9.1.3 Vytváření a úprava skupin

Vytvoření skupiny provedeme pomocí příkazu **sudo groupadd [možnosti] [jméno skupiny]**. Pokud chceme přidat uživatele do skupiny, je vhodné si nejdříve ověřit, že daná skupina má oprávnění, které chceme danému uživateli přiřadit. Pro přidání daného uživatele využijeme příkazu **sudo usermod -aG [jméno skupiny] [uživatelské jméno]**. Možnost **-aG** přidá uživatele do konkrétní již existující skupiny, aniž by uživatele odstranila z jeho aktuální skupiny. Možnost **-G** bez argumentu **-a** přidá uživatele do jmenované skupiny a pokud je členem jiné skupiny nezahrnuté v příkazu, bude z ní automaticky smazán. [85]

Pokud bychom chtěli změnit skupinu, využijeme příkazu **sudo groupmod [možnosti] [jméno skupiny]**. Takto můžeme například změnit název skupiny či GID skupiny.[85]

---

[29] Malé L.

Před odstraněním skupiny je třeba zkontrolovat, že již neexistují soubory či uživatelé spojení s danou skupinou, jelikož by mohli vyvolat problémy s přístupy k účtům či souborům. Pokud smažeme skupinu, ale ne její soubory, při vytvoření skupiny se stejným GID jako měla odstraněná skupina se soubory připojí k této skupině a budou si je moci zobrazit pouze uživatelé této nové skupiny. Pro odstranění skupiny využíváme příkazu **sudo groupdel [jméno skupiny]**.

9.1.4 Oprávnění skupiny uživatelů

V Linuxu má každý účet či skupina přiřazená jistá oprávnění. Tato oprávnění definují, co daný uživatel či uživatelé ve skupině mohou provádět. Oprávnění se vztahují na soubory a adresáře, a tím pádem také na příkazy, které může daný uživatel použít. Existují tři druhy oprávnění, a to čtení (r), zápis (w) a spuštění souboru (x). Oprávnění souboru můžeme definovat dvěma způsoby, a to absolutním či symbolickým stylem. Absolutní styl používá čísla pro specifikaci oprávnění. Čtení má hodnotu 4, zápis 2 a spuštění 1. Každý typ identity je definován součtem hodnot oprávnění identity. Jsou tři typy identity, a to uživatel, skupina a ostatní. Při použití symbolického stylu používáme písmena **r**, **w** a **x** jako typy oprávnění a identity mají symbol **u** (user), **g** (group) a **o** (other). Při použití symbolického stylu také používáme operátory '+','-' a '='. '+' přidává dané oprávnění uživateli. '-' uživateli odstraňuje oprávnění a '=' nastavuje daná oprávnění. Pro přiřazení oprávnění k souboru využíváme příkazu **chmod [oprávnění] [název souboru]**. Příkaz pro nastavení oprávnění uživatele na čtení a zápis, pro skupinu na zápis a pro ostatní na spuštění souboru vypadá v absolutním stylu takto: **chmod 621 [název souboru]**, v symbolickém stylu takto: **chmod u=rw, g=w, o=x [název souboru]**.[85]

Další oprávnění jsou **setuid** a **setgid**. **Setuid** je oprávnění, které může být přiřazeno ke spustitelným souborům v operačních systémech Unix. Toto oprávnění nastaví spuštění souboru s oprávněními vlastníka souboru, ne s oprávněními uživatele, jenž jej spouští. Toto můžeme využít pro provádění příkazů s vyšším oprávněním než naším, například změnu hesla nebo pro přístup k běžně nepřístupným souborům. Pokud chceme v oprávnění souboru najít, zda má nastavené **setuid** oprávnění, hledáme (**s**) pro **setuid** oprávnění místo (**x**) pro spustitelnou aplikaci. Pro absolutní styl **setuid** patří hodnota 4, která je svojí vlastní identitou před dalšími tedy

například chmod 4264 nastaví setuid s hodnotou 4, 2 je oprávnění pro uživatele, 6 je pro skupinu a 4 je pro ostatní uživatele. **Setgid** je velmi podobné **setuid**. Toto nastavení opravňuje uživatele ke spuštění souboru pomocí oprávnění skupiny vlastníka souboru. Použitím **setgid** u adresářů nastaví stejná skupinová práva nových souborů nebo adresářů jako u mateřského adresáře. Toto oprávnění je využíváno při skupinové práci na souborech, jelikož tím zajistíme, že všechny soubory vytvořené v těchto adresářích budou mít stejná skupinová práva. Pro zobrazení **setgid** oprávnění se podíváme na nastavení skupiny u oprávnění souboru a pokud je **setgid** nastaveno, uvidíme hodnotu (**s**). Hodnota pro absolutní styl je 2. Využití **setuid** a **setgid** má i bezpečnostní rizika, jelikož potenciálně můžeme uživateli zpřístupnit root přístup k systému, a nejen tato oprávnění k danému souboru. [86]

Mezi další důležité oprávnění také patří **sudo**. **Sudo** je příkaz, jenž nám dovolí provádět příkazy s privilegii jiného uživatele, ale stále pomocí našeho účtu a většinou se používá pro používání příkazu s privilegii root uživatele. Sudo příkazy by měl uživatel používat, pouze pokud je k tomu oprávněn a má důvod tyto příkazy použít. Toto provedeme dvěma různými příkazy za použití uživatelského účtu se **sudo** privilegii, a to pomocí přidání uživatele do **sudo** skupiny (**sudo usermod -aG sudo [jméno uživatele]**) či pomocí úpravy *Sudoers* souboru. Upravení souboru vykonáme pomocí příkazu **sudo visudo**[30] a zde přidáme nový řádek na konec seznamu, který obsahuje **[název uživatele] ALL= (ALL) ALL**. Pokud bychom nechtěli, aby uživatel měl práva na provádění všech příkazů, můžeme při úpravě souboru definovat, které příkazy může uživatel vykonávat. Toto provedeme zadáním tohoto obsahu řádku **[uživatelské jméno] ALL=(ALL) [seznam příkazů oddělený čárkami]** místo předchozího obsahu. Při využití **sudo** příkazu se nás systém zeptá na naše heslo. Dá se i nastavit, aby se systém neptal vždy a byla nastavena určitá časová prodleva od posledního dotazu či celkové zrušení dotazování se, ale je *silně* nedoporučováno toto nastavit. [88]

---

[30] Příkaz **visudo** používáme pro editaci souboru Sudoers, jelikož kontroluje syntax před uložením, a tím zabraňuje nepoužitelnosti tohoto souboru a také možnosti vytvoření nepoužitelného systému. [87]

9.1.5 Odkazy

Odkazy jsou ukazatele na soubory nebo adresáře v jiném adresáři a ulehčují nám přístup k jinému souboru. Existují dva druhy odkazů, a to symbolické a pevné, nebo měkké a tvrdé. Symbolické odpovídají měkkým odkazům a pevné tvrdým odkazům.

Pevné odkazy

Každý pevný odkaz má stejnou inode hodnotu jako zdrojový soubor, a tedy odkazují na stejnou fyzickou lokaci souboru. To také znamená, že mají stejná metadata, a proto se při přesunutí nebo odstranění zdrojového souboru pořád dá použít daný odkaz. Pevný odkaz nezabírá žádné místo navíc a nemůžeme je vytvořit pro soubory v jiných souborových systémech. Pro vytvoření využijeme příkaz **ln [zdrojový soubor] [název pevného odkazu]**. [89]

Symbolické odkazy

Symbolický odkaz nepřesměrovává na stejnou fyzickou lokaci souboru, ale odkazuje na zdrojový soubor. Při přesunu či odstranění zdrojového souboru se tedy symbolický odkaz pokazí, protože odkazuje na neexistující lokaci. Narozdíl od pevných odkazů také můžou odkazovat na soubory v jiných souborových systémech. Pro vytvoření využijeme příkaz **ln -s [cesta ke zdrojovému souboru] [cesta k symbolickému odkazu]**. [89]

9.1.6 Firewall konfigurace

Firewall sleduje a řídí příchozí a odchozí síťový provoz. Podle definovaných bezpečnostních pravidel může povolit či blokovat daný provoz.[90] Linuxové firewally používají Netfilter subsystém, který filtruje pakety a řídí jejich provoz. Dále je využíváno nástroje *iptables*, který je používán na nastavování pravidel firewallu. Když přijde paket na server, je předán Netfilter, který rozhodne o přijetí či odmítnutí na základě pravidel z *iptables*. Na konfiguraci firewallu se standardně používá nástroj *ufw* (uncomplicated firewall). Pokud uživatel ufw prozatím na svém zařízení nepoužil, je třeba jej zapnout pomocí příkazu **sudo ufw enable**. Pokud chceme povolit otevření portu, použijeme příkaz **sudo ufw allow [číslo portu[/číslo protokolu]]** a příkazem **sudo ufw deny [číslo portu[/číslo protokolu]]** port

zavřeme. Pokud se rozhodneme změnit pravidlo a chceme jej smazat, aby se nepoužívalo, využijeme příkazu **sudo ufw delete [pravidlo, které nechceme používat]**. Například pro smazání pravidla uzavření portu 22 využijeme příkazu **sudo ufw delete deny 22**. Pokud bychom chtěli povolit přístup z určitých hostitelů na port, použijeme příkaz **sudo ufw allow from [IP adresa] to [cílová IP adresa nebo any] port [číslo portu[/číslo protokolu]]**. [91]

Po přidání či změnu pravidel znovu načteme firewall pro provedení změn pomocí příkazu **sudo ufw reload**. Pokud si s novými změnami nejsme jisti, lze je vyzkoušet pomocí tzv. dry run. Příkaz **sudo ufw --dry-run allow [pravidlo]** nám simuluje situaci, kdy by bylo dané pravidlo provedeno, a poté se můžeme podle výstupu rozhodnout, zda dané pravidlo vytvoříme či upravíme. Pro zobrazení nastavených pravidel můžeme použít příkaz **sudo ufw status**.[91]

9.2 *Praktická úloha*

V této praktické úloze provedeme vytvoření uživatelského účtu, který pojmenujeme **uzivatel** a vytvoříme mu soubory. Poté daný uživatelský účet smažeme a vyhledáme soubory, jejichž vlastníkem je smazaný uživatel a také je smažeme.
Kroky postupu:

1. Vytvoříme uživatele jménem **uživatel**
2. Přepneme se do **uzivatel** účtu
3. Vytvoříme prázdné soubory s názvy **soubor.txt**, **soubory.txt**
4. Přepneme se zpět do **administrátor** účtu
5. Smažeme uživatele
6. Prohledáme všechny soubory pro nalezení souborů, jehož autorem byl **uživatel**
7. Smažeme vyhledané soubory

9.2.1 Návod

Nový uživatelský účet vytvoříme a pojmenujeme **uzivatel** příkazem **sudo useradd uzivatel**. Přepneme se do **uzivatel** účtu pomocí příkazu **su uzivatel**. Vytvoříme prázdné soubory, které můžeme ale nemusíme naplnit pomocí příkazu **vi soubor.txt** a **vi soubory.txt**. Můžeme jich klidně vytvořit i více. Přepneme se zpět do **administrátor** účtu pomocí příkazu **su root**.
Dále uživatelský účet smažeme příkazem **sudo userdel uživatel**. A pro prohledání systému využijeme příkazu **sudo find / -user uzivatel**. Poté tyto soubory smažeme příkazem **sudo rm soubor.txt soubory.txt**.
Nebo se smazání souborů dá provést rovnou při mazání uživatele příkazem **sudo userdel -r uzivatel**.

9.3 *Otázky na ověření znalostí*

1 Jaké jsou typy skupin a uživatelů?

   a) V Linuxu existují dva typy skupin, a to primární a sekundární. Primární skupiny jsou uživateli přiřazeny při vytvoření a každý uživatel musí patřit alespoň do jedné. Do sekundárních skupin uživatel nemusí patřit, ale může patřit až do 15 skupin. Dále existují dva typy uživatelů, a to root a běžný uživatel. Root uživatel má všechny privilegia, zatímco běžný uživatel má limitovaný přístup k souborům a operacím. Někteří běžní uživatelé mohou využívat **sudo** privilegií.

2 K čemu jsou využívána **sudo** privilegia?

   a) **Sudo** privilegia jsou využívána k provedení příkazu pomocí práv root uživatele.

3 Jaké jsou využití příkazu **passwd**?

   a) Příkaz **passwd** se využívá pro správu hesel. Dá se pomocí něho změnit heslo, úprava doby platnosti hesla či zablokování a odblokování účtů.

4 Jaké jsou druhy oprávnění?

   a) Existují tři druhy oprávnění vztahujících se k souborům a adresářům, a to jsou read, write a execute nebo-li čtení, zápis a spuštění souboru. Tato oprávnění jsou u souborů a adresářů dále rozdělena podle kategorií přístupu na uživatel, skupina a ostatní. Dále existují oprávnění **setuid** a **setgid**. **Setuid** může být přiřazeno ke spustitelným souborům a nastavuje spuštění daného souboru s oprávněním vlastníka, a ne s oprávněními uživatele, který ho spouští. **Setgid** nastavuje spuštění daného souboru s oprávněním skupiny vlastníka souboru. Dalším důležitým oprávněním je **sudo**. Toto oprávnění nám dovolí provádět příkazy s privilegii jiného uživatele, kterým většinou bývá root.

5 Jaké jsou styly zapsání oprávnění a jaký je mezi nimi rozdíl?

   a) Styly zapsání oprávnění jsou dva, a to symbolické a absolutní. Absolutní styl využívá čísla pro specifikaci oprávnění a dané hodnoty sčítá pro každou kategorii přístupu. Čtení má hodnotu 4, zápis 2 a spuštění 1. Symbolický styl používá písmena r(čtení),w(zápis) a x(spuštění) jako hodnoty pro dané

oprávnění. Daná písmena jsou zapsaná vedle sebe v jednom řetězci pro každou kategorii přístupu.

6 Jak zjistíme a změníme oprávnění?

a) Oprávnění změníme pomocí příkazu **chmod [oprávnění] [název souboru]**, kde zapíšeme oprávnění pomocí symbolického či absolutního stylu dosazeného do „rovnice“. Pro zjištění oprávnění použijeme například příkazu **ls -l [název souboru]**.

7 Jaký je rozdíl mezi symbolickými a pevnými odkazy?

a) Symbolický odkaz odkazuje na lokaci daného souboru, zatímco pevný odkaz odkazuje na fyzickou lokaci souboru, s kterým sdílí inode hodnotu a metadata. Rozdíl mezi nimi je poznat například při přesunutí zdrojového souboru, jelikož symbolický odkaz přestane fungovat, ale pevný odkaz nadále bude funkční.

8 Na co je využíván nástroj **iptables**?

a) Nástroj **iptables** je používán na nastavování pravidel firewallu. Tyto pravidla rozhodují například o přijetí či zamítnutí paketů.

Závěr

Operační systém Linux a jeho distribuce Ubuntu má velmi rozsáhlé možnosti nejen pro začátečníky. Není tedy jednoduché obsáhnout veškerý výklad práce s tímto systémem do jednoho návodu vhodného pro začátečníky. Cílem této bakalářské práce bylo představit začátečníkům práci s Ubuntu, což bylo docíleno pomocí laboratorních úloh. Úloha 1 se zabírala instalací a úvodním nastavením Ubuntu. Úloha 2 se zaobírala konfigurací systému a hardwaru, úloha 3 probírala konfiguraci sítě a zabezpečení systému. Úloha 4 probírala práci se soubory a v poslední úloze byla probraná administrace systému. Tyto úlohy neprobraly všechno potřebné a proto je žádoucí, aby se po prostudování práce uživatel dále sám vzdělával. Ve výkladové části a teoretických částech každé laboratorní úlohy je rozepsán postup vhodný k vykonání daného úkonu a popřípadě jeho další alternativy. Tyto postupy mohou být v budoucnosti zastaralé. Naplnění cíle práce bylo formátováno do laboratorních úloh, které vysvětlují vše potřebné ke splnění teoretických otázek a praktických úloh uvedených na závěru každé laboratorní úlohy. Praktická úloha obsahuje látku z většiny kapitol dané laboratorní úlohy, případně znalosti z předešlých laboratorních úloh pro spojení všech již načerpaných vědomostí.

Tuto práci by bylo možné dále rozšířit rozpracováním složitějších úkonů a příkazů. Rozšíření by mohlo být vedeno ve stejném formátu laboratorních úloh s teoretickými otázkami a praktickými úlohami na jejich závěru. Tento formát by byl přístupný začátečníkům, čímž by pokračovala jejich vzdělávací cesta s operačním systémem Linux.

Seznam použité literatury

1. *Master Linux Operating Systems With NDG Linux II Course*. (2024, únor 29). Networking Academy. https://www.netacad.com/courses/os-it/ndg-linux-II
2. *Advance Your Skills With NDG Linux I Course*. (2024, únor 29). Networking Academy. https://www.netacad.com/courses/os-it/ndg-linux-I
3. *In Depth Training With NDG Linux Essentials Course*. (2024, únor 29). Networking Academy. https://www.netacad.com/courses/os-it/ndg-linux-essentials
4. History of Linux. (2023, únor 12). *GeeksforGeeks*. https://www.geeksforgeeks.org/linux-history/
5. *Difference between Linux distributions*. (b.r.). ComputerNetworkingNotes. Získáno 27. únor 2024, z https://www.computernetworkingnotes.com/linux-tutorials/difference-between-linux-distributions.html
6. meilinaeka. (2023, duben 28). Linux Operating System: History, Functions, Advantages, and Disadvantages. *Direktorat Pusat Teknologi Informasi*. https://it.telkomuniversity.ac.id/en/linux-operating-system/
7. *Android | Definition, History, & Facts | Britannica*. (2024, únor 26). https://www.britannica.com/technology/Android-operating-system
8. Machkovech, S. (2022, únor 25). *A brief tour of the Steam Deck's Linux implementation*. Ars Technica. https://arstechnica.com/gaming/2022/02/linux-on-steam-deck-what-you-need-to-know-what-currently-works/
9. Pablinux. (2019, duben 1). Linux je králem, pokud jde o Smart TV. A půjde to dál. *Linux Adictos*. https://www.linuxadictos.com/cs/Linux-je-

kr%C3%A1lem%2C-pokud-jde-o-inteligentn%C3%AD-televizi%2C-a-st%C3%A1le-jde-je%C5%A1t%C4%9B-d%C3%A1l.html

10. Sharma, D. (2022, červenec 22). *Understanding the Linux Kernel Versioning Scheme*. MUO. https://www.makeuseof.com/how-are-linux-kernel-versions-formed/
11. *What is Patch Management in Linux? | Best Features and Benefits*. (b.r.). ITarian. Získáno 27. únor 2024, z https://www.itarian.com/itsm/what-is-patch-management-in-linux.php
12. *Applying Patches To The Linux Kernel—The Linux Kernel documentation*. (b.r.). Získáno 27. únor 2024, z https://www.kernel.org/doc/html/next/process/applying-patches.html
13. *About the Ubuntu project*. (b.r.). Ubuntu. Získáno 27. únor 2024, z https://ubuntu.com/about
14. Molochko, A. (2023, květen 15). *Why Is Linux a Monolithic Kernel? | Baeldung on Linux*. https://www.baeldung.com/linux/monolithic-kernel
15. *What is the difference between the kernel and user spaces?* (b.r.). Educative. Získáno 28. únor 2024, z https://www.educative.io/answers/what-is-the-difference-between-the-kernel-and-user-spaces
16. *Central processing unit (CPU) | Definition & Function | Britannica*. (2024, únor 26). https://www.britannica.com/technology/central-processing-unit
17. *X86: Everything You Need To Know About X86 | Lenovo US*. (b.r.). Získáno 1. březen 2024, z https://www.lenovo.com/us/en/glossary/x86/

18. Singh, R. (b.r.). *Understanding the Boot process—BIOS vs UEFI*. Získáno 18. listopad 2023, z https://linuxhint.com/understanding_boot_process_bios_uefi/

19. Hoffman, C. (2014, září 22). *How to Configure the GRUB2 Boot Loader's Settings*. How-To Geek. https://www.howtogeek.com/196655/how-to-configure-the-grub2-boot-loaders-settings/

20. *RecoveryMode—Ubuntu Wiki*. (b.r.). Získáno 10. leden 2024, z https://wiki.ubuntu.com/RecoveryMode

21. Both, D. (b.r.). *Understanding systemd at startup on Linux | Opensource.com*. Získáno 22. leden 2024, z https://opensource.com/article/20/5/systemd-startup

22. *Install Ubuntu desktop*. (b.r.). Ubuntu. Získáno 18. listopad 2023, z https://ubuntu.com/tutorials/install-ubuntu-desktop

23. McKay, D. (2021, červen 9). *How to Install Google Chrome on Ubuntu Linux*. How-To Geek. https://www.howtogeek.com/731805/how-to-install-google-chrome-on-ubuntu-linux/

24. How to install Opera Web Browser on Ubuntu. (2022, červen 1). *GeeksforGeeks*. https://www.geeksforgeeks.org/how-to-install-opera-web-browser-on-ubuntu/

25. khess. (2019, srpen 20). *An introduction to the vi editor*. Red Hat, Inc. https://www.redhat.com/sysadmin/introduction-vi-editor

26. *What are man pages and why are they important to your Linux education?* (b.r.). ZDNET. Získáno 25. únor 2024, z https://www.zdnet.com/article/what-are-man-pages-and-why-are-they-important-to-your-linux-education/

27. *Install Docker Engine on Ubuntu*. (100n. l., 37:01 + +0100). Docker Documentation. https://docs.docker.com/engine/install/ubuntu/

28. user3133300. (2015, únor 18). *Identifying CPU from lscpu command?* [Forum post]. Super User. https://superuser.com/q/879644

29. Batool, S. W. (b.r.). *How to Use Linux lscpu Command Tutorial*. Získáno 21. leden 2024, z https://linuxhint.com/lscpu-command/

30. baeldung. (2021, červenec 17). *The Most Common Flags in /proc/cpuinfo with Examples | Baeldung on Linux*. https://www.baeldung.com/linux/proc-cpuinfo-flags

31. */Proc/cpuinfo file explained – The Geek Diary*. (b.r.). Získáno 26. březen 2024, z https://www.thegeekdiary.com/proccpuinfo-file-explained/

32. *How to Use the Linux Free Command*. (b.r.). Získáno 21. leden 2024, z https://www.turing.com/kb/how-to-use-the-linux-free-command

33. baeldung. (2022, srpen 9). *The /proc/meminfo File in Linux | Baeldung on Linux*. https://www.baeldung.com/linux/proc-meminfo

34. *Create a Partition in Linux—A Step-by-Step Guide | DigitalOcean*. (b.r.). Získáno 21. leden 2024, z https://www.digitalocean.com

35. Hoffman, C., & Lewis, N. (2017, únor 9). *What's the Difference Between GPT and MBR When Partitioning a Drive?* How-To Geek. https://www.howtogeek.com/193669/whats-the-difference-between-gpt-and-mbr-when-partitioning-a-drive/

36. Both, D. (b.r.). *An introduction to Linux's EXT4 filesystem | Opensource.com*. Získáno 23. leden 2024, z https://opensource.com/article/17/5/introduction-ext4-filesystem

37. Matador, B. (b.r.). *What is an inode and what are they used for?* Získáno 23. leden 2024, z https://www.bluematador.com/blog/what-is-an-inode-and-what-are-they-used-for

38. *XFS vs Ext4: Which One Is Better? - MiniTool Partition Wizard*. (2021, duben 27). MiniTool. https://www.partitionwizard.com/partitionmanager/xfs-vs-ext4.html

39. Jebavý, J. (b.r.). *Souborový systém Btrfs: Vlastnosti a výhody moderního ukládání dat*. Root.cz. Získáno 23. leden 2024, z https://www.root.cz/clanky/souborovy-system-btrfs-vlastnosti-a-vyhody-moderniho-ukladani-dat/

40. *Linux File Systems for Windows: Use EXT4 / XFS / Btrfs On Windows*. (b.r.). Získáno 23. leden 2024, z https://www.phoronix.com/news/Linux-File-Systems-for-Windows

41. UK, D. (2017, červen 5). Best Drive Format For Multiple Operating Systems. *Ditly*. https://ditly.uk/best-drive-format-for-multiple-operating-systems/

42. *Allow Users in fstab File to Read and Write to a Partition | Baeldung on Linux*. (2022, červenec 28). https://www.baeldung.com/linux/fstab-file-users-read-write-partition

43. *How to mount disk and partition in Linux*. (b.r.). Získáno 2. duben 2024, z https://www.simplified.guide/linux/disk-mount

44. baeldung. (2022, listopad 12). *Swap Partition vs Swap File | Baeldung on Linux*. https://www.baeldung.com/linux/swap-file-partition

45. *Oracle Linux 6: Administrator's Guide*. (b.r.). Získáno 21. leden 2024, z https://docs.oracle.com/en/operating-systems/oracle-linux/6/admin/swap-partitiion-create.html
46. *Oracle Linux 6: Administrator's Guide*. (b.r.). Získáno 21. leden 2024, z https://docs.oracle.com/en/operating-systems/oracle-linux/6/admin/swap-create-use.html
47. ClearAI. (2023, březen 23). *The Ultimate Guide To Logging: What It Is And Why You Need It*. ClearInsights. https://clearinsights.io/blog/the-ultimate-guide-to-logging-what-it-is-and-why-you-need-it/
48. *How To Monitor System Authentication Logs on Ubuntu | DigitalOcean*. (b.r.). Získáno 26. únor 2024, z https://www.digitalocean.com/community/tutorials/how-to-monitor-system-authentication-logs-on-ubuntu
49. *Viewing and monitoring log files*. (b.r.). Ubuntu. Získáno 26. únor 2024, z https://ubuntu.com/tutorials/viewing-and-monitoring-log-files
50. Both, D. (b.r.). *How I use cron in Linux | Opensource.com*. Získáno 25. leden 2024, z https://opensource.com/article/17/11/how-use-cron-linux
51. *How To Use IPRoute2 Tools to Manage Network Configuration on a Linux VPS | DigitalOcean*. (b.r.). Získáno 21. leden 2024, z https://www.digitalocean.com
52. *The /etc/resolv.conf File | Baeldung on Linux*. (2023, březen 11). https://www.baeldung.com/linux/etc-resolv-conf-file
53. www.benes-michl.cz, B. & M. (b.r.). *Co je to DNS a jak funguje? | Airwaynet.cz - Internet na doma v Praze*. Získáno 26. únor 2024, z https://www.airwaynet.cz/co-je-to-dns-a-jak-funguje/

54. Computer Network | AAA (Authentication, Authorization and Accounting). (2018, červen 26). *GeeksforGeeks*. https://www.geeksforgeeks.org/computer-network-aaa-authentication-authorization-and-accounting/

55. Bhatt, M. (b.r.). *RADIUS Server (RADIUS Authentication) and How it Works*. Získáno 26. únor 2024, z https://www.foxpass.com/blog/radius-server-and-how-it-works

56. *SSH Tutorial (Linux)*. (b.r.). Zentrum Für Astronomie. Získáno 21. leden 2024, z https://zah.uni-heidelberg.de/it-guide/ssh-tutorial-linux

57. *SSH Agent Explained*. (b.r.). Smallstep. Získáno 29. únor 2024, z https://www.smallstep.com/

58. *Man-in-the-Middle*. (b.r.). [Page]. ENISA. Získáno 21. leden 2024, z https://www.enisa.europa.eu/topics/incident-response/glossary/man-in-the-middle

59. *The GNU Privacy Handbook*. (b.r.). Získáno 21. leden 2024, z https://www.gnupg.org/gph/en/manual/book1.html

60. File globbing in Linux. (2018, duben 15). *GeeksforGeeks*. https://www.geeksforgeeks.org/file-globbing-linux/

61. *Regex basics*. (b.r.). Ubuntu. Získáno 25. únor 2024, z https://ubuntu.com/blog/regex-basics

62. Megida, D. (b.r.). *Regex vs Globbing | Differences and Similarities—Dillion's Blog*. Získáno 25. únor 2024, z https://dillionmegida.com/p/regex-vs-glob-patterns/

63. Ls Command in Linux. (2017, červen 4). *GeeksforGeeks*. https://www.geeksforgeeks.org/ls-command-in-linux/

64. File command in Linux with examples. (2019, únor 18). *GeeksforGeeks*. https://www.geeksforgeeks.org/file-command-in-linux-with-examples/

65. How to Create an Empty File in Linux | Touch Command. (2018, prosinec 20). *GeeksforGeeks*. https://www.geeksforgeeks.org/touch-command-in-linux-with-examples/

66. Locate command in Linux with Examples. (2019, leden 22). *GeeksforGeeks*. https://www.geeksforgeeks.org/locate-command-in-linux-with-examples/

67. *Invoking updatedb (GNU Findutils 4.9.0)*. (b.r.). Získáno 29. únor 2024, z https://www.gnu.org/software/findutils/manual/html_node/find_html/Invoking-updatedb.html#Invoking-updatedb

68. *Excluding Directories from „locate" (updatedb) [JoelDare.com]*. (b.r.). Získáno 29. únor 2024, z https://www.joeldare.com/wiki/linux:excluding_directories_from_locate_updatedb

69. *The Linux cp Command – How to Copy Files in Linux*. (2022, červen 6). freeCodeCamp.Org. https://www.freecodecamp.org/news/the-linux-cp-command-how-to-copy-files-in-linux/

70. *mv Command in Ubuntu: 7 Practical Examples*. (2022, červenec 5). Learn Ubuntu. https://learnubuntu.com/mv-command/

71. Rm command in Linux with examples. (2018, leden 3). *GeeksforGeeks*. https://www.geeksforgeeks.org/rm-command-linux-examples/

72. How to Create Directory in Linux | mkdir Command. (2018, prosinec 12). *GeeksforGeeks*. https://www.geeksforgeeks.org/mkdir-command-in-linux-with-examples/

73. *How to Delete a File or Directory in Linux – Command to Remove a Folder and its Contents*. (2023, květen 4). freeCodeCamp.Org. https://www.freecodecamp.org/news/how-to-delete-a-file-or-directory-in-linux/

74. How to Compress Files in Linux | Tar Command. (2017, říjen 4). *GeeksforGeeks*. https://www.geeksforgeeks.org/tar-command-linux-examples/

75. *How Zip Files Work: A Guide*. (b.r.). Získáno 26. leden 2024, z https://experience.dropbox.com/resources/what-is-a-zip-file

76. *Similarities and differences between gzip and bzip2*. (b.r.). ComputerNetworkingNotes. Získáno 26. leden 2024, z https://www.computernetworkingnotes.com/linux-tutorials/similarities-and-differences-between-gzip-and-bzip2.html

77. baeldung. (2020, červen 23). *Using xz Compression in Linux | Baeldung on Linux*. https://www.baeldung.com/linux/xz-compression

78. Gunzip command in Linux with examples. (2019, březen 6). *GeeksforGeeks*. https://www.geeksforgeeks.org/gunzip-command-in-linux-with-examples/

79. *HOW TO MANAGE USERS AND GROUPS IN LINUX OPERATING SYSTEM? CONTINUED | LinkedIn*. (b.r.). Získáno 24. leden 2024, z https://www.linkedin.com/pulse/how-manage-users-groups-linux-operating-system-continued-assefa/

80. baeldung. (2022, červen 6). *Primary and Secondary Groups in Linux | Baeldung on Linux*. https://www.baeldung.com/linux/primary-vs-secondary-groups

81. Least Privilege. (b.r.). *CyberArk*. Získáno 24. leden 2024, z https://www.cyberark.com/what-is/least-privilege/

82. *What is a mail spool? - Knowledgebase—Advantagecom Networks, Inc.* (b.r.). Získáno 24. leden 2024, z https://cbp.speedingbits.com/billing/index.php?rp=/knowledgebase/23/What-is-a-mail-spool.html

83. How to Delete User in Linux | userdel Command. (2019, březen 27). *GeeksforGeeks*. https://www.geeksforgeeks.org/userdel-command-in-linux-with-examples/

84. How to Change User Password in Linux | passwd Command. (2018, prosinec 27). *GeeksforGeeks*. https://www.geeksforgeeks.org/passwd-command-in-linux-with-examples/

85. dgarn. (2020, listopad 26). *How to manage Linux permissions for users, groups, and others*. Red Hat, Inc. https://www.redhat.com/sysadmin/manage-permissions

86. SetUID, SetGID, and Sticky Bits in Linux File Permissions. (2019, srpen 5). *GeeksforGeeks*. https://www.geeksforgeeks.org/setuid-setgid-and-sticky-bits-in-linux-file-permissions/

87. *How To Edit the Sudoers File | DigitalOcean*. (b.r.). Získáno 25. únor 2024, z https://www.digitalocean.com/community/tutorials/how-to-edit-the-sudoers-file

88. baeldung. (2022, duben 12). *How to Give Sudo Privileges to a User in Linux | Baeldung on Linux*. https://www.baeldung.com/linux/sudo-privileges-user

89. Soft and Hard links in Unix/Linux. (2017, září 26). *GeeksforGeeks*. https://www.geeksforgeeks.org/soft-hard-links-unixlinux/

90. *What Is a Firewall?* (b.r.). Cisco. Získáno 26. únor 2024, z https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html

91. *Security—Firewall*. (b.r.). Ubuntu. Získáno 26. únor 2024, z https://ubuntu.com/server/docs/security-firewall

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**Michaela Kratochvílová**
Studium:	I2100231
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Praktické úlohy konfigurace a správy GNU/Linux Ubuntu**
Název bakalářské práce AJ:	Practical tasks about configuration and administration of GNU/Linux Ubuntu

Cíl, metody, literatura, předpoklady:

Cílem bakalářské práce je vytvořit sadu řešených pokročilých úloh zaměřených na konfiguraci a správu distibuce GNU/Linux Ubuntu. Práce propojí teoretické principy přístupů ke konfiguraci a správě Ubuntu, které rozšíří o vhodně zvolené a podrobně řešené praktické úlohy.

CISCO Networking Academy Program, NDG Linux Essential.

CISCO Networking Academy Program, NDG Linux I.

CISCO Networking Academy Program, NDG Linux II.

Zadávající pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	doc. Mgr. Josef Horálek, Ph.D.

Datum zadání závěrečné práce: 15.10.2021