UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2024 Martin Mandík

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Webová aplikace pro výuku Bash

Bakalářská práce

2024 Martin Mandík

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin Mandík**
Osobní číslo:	**I21187**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Webová aplikace pro výuku Bash**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je vytvoření webové aplikace pro výuku skriptovacího jazyka bash. V teoretické části budou představeny základní a pokročilé koncepty jazyka bash. V praktické části bude realizovaná webová aplikace umožňující výuku skriptovacího jazyka. Výuková aplikace bude obsahovat následující témata:
Základní práce s příkazovým řádkem
Práce se souborovým systémem
Manipulace s textem (výstupy, zabezpečení vstupu)
Proměnné, datové typy
Podmínky, cykly
Pole, funkce
Správa procesů
Plánování (zautomatizování) úloh
Tipy pro práci s bashem

Rozsah pracovní zprávy: **30**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

KAMENÍK, Pavel. Příkazový řádek v Linuxu: praktická řešení. Brno: Computer Press, 2011. ISBN 978-80-251-2819-0
BILLEMONT, Maarten. Bash Guide. Online. 1. Billemont. ISBN 978-19-806-8960-7. Dostupné z: https://lhunath.com/. [cit. 2023-10-05].

Vedoucí bakalářské práce: **Ing. Soňa Neradová, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2024

Prohlašuji:

Práci s názvem „*webová aplikace pro výuku Bash“* jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 06. 05. 2024

Martin Mandík

PODĚKOVÁNÍ

Rád bych poděkoval své vedoucí bakalářské práce Ing. Soně Neradové, Ph.D. za poskytnutí odborných rad a připomínek v průběhu zpracování této bakalářské práce. Také bych chtěl poděkovat rodině a přátelům za podporu v průběhu studia.

ANOTACE

Tato práce se zabývá skriptovacím jazykem Bash, návrhem a vytvoření webové aplikace pro výuku příkazů tohoto jazyka. Teoretická část se věnuje základním principům, od způsobu vyhledání informací pomocí manuálových stránek, základních příkazů, příkazů souborového systému, příkazů pro zpracování textu, po příkazy vstupů, výstupů a rour. Dále teoretická část obsahuje popis vytvoření skriptu, použití proměnných, příkazů pro řízení toku nebo funkcí. V praktické části této práce jsou popsány použité nástroje použité pro implementaci webové aplikace a popsány jednotlivé části webové aplikace.

KLÍČOVÁ SLOVA

Bash, skriptování, příkazy, automatizace, GNU, otevřený zdrojový kód

TITLE

Bash Tutorial Web App

ANNOTATION

This thesis deals with the Bash scripting language, designing and creating a web application for learning the commands of this language. The theoretical part covers the basic principles, from how to find information using manual pages, basic commands, file system commands, text processing commands, to input, output and router commands. In addition, the theoretical section includes descriptions of script creation, the use of variables, flow control commands, or functions. The practical part of this thesis describes the tools used to implement the web application and describes the different parts of the web application.

KEYWORDS

Bash, scripting, commands, automatization, GNU, open source

OBSAH

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

SEZNAM ZKRATEK A ZNAČEK

Bash	Bourne Again Shell
CSS	Cascading Style Sheets
GNU	GNU’s Not Unix
HTML	HyperText Markup Language
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
LTS	Long Term Support
SQL	Structured Query Language
TCP	Transmission Control Protocol
XML	eXtensible Markup Language
YAML	YAML Ain’t Markup Language

TERMINOLOGIE

Shell – Rozhraní sloužící k interakci s operačním systémem, ovládá se pomocí zadávaných příkazů

FIFO – First In, First Out, jedná se o metodu zpracování dat, data jsou zpracována v pořadí, v jakém přišla

JVM – Java Virtual Machine, virtuální stroj pro běh programů zkompilovaných do Java bytecodu

WYSIWIG – What You See Is What You Get, software, který zobrazuje konečný výsledek během úpravy obsahu, při jeho použití není potřeba pracovat například přímo se značkovacím jazykem

API – Application Programming Interface, rozhraní pro komunikaci a výměnu dat různých aplikací mezi sebou

POSIX – Portable Operating System Interface, skupina standardů zajišťující kompatibilitu mezi různými operačními systémy

ÚVOD

V dnešní době je připojeno k internetu stále více zařízení. Téměř každé zařízení minimálně vyžaduje prvotní konfiguraci a později alespoň minimální údržbu. Při narůstajícím množství zařízení je ruční správa každého jednotlivého zařízení velice náročná na lidské zdroje. Možným řešením je konfiguraci a správu těchto zařízení co nejvíce zautomatizovat, například pomocí skriptů. Velké množství zařízení používá nějaký z Unixových operačních systému, dnes nejčastěji Linux, kde již bývá Bash předinstalován.

Hlavním cílem této bakalářské práce je vytvoření webové aplikace, která bude sloužit k výuce skriptovacího jazyka Bash. Tato aplikace provede uživatele od úplně základních příkazů, až po vytváření vlastních skriptů. Zároveň bude mít uživatel v aplikaci dostupný terminál s Bashem, kde si bude moci příkazy procvičovat.

Teoretická část práce se věnuje představení základních principů Bashe, základních příkazů, příkazů souborového systému, vstupů až po skriptování. Následně jsou popsány technologie použité k implementaci aplikace a popis webové aplikace.

1 PŘEDSTAVENÍ BASHE

Bourne Again Shell, zkráceně Bash je příkazový jazyk a Unixový shell. Vznikl jako svobodná náhrada Bourne Shellu, zkráceně sh. Dnes je jedním z nejrozšířenějších a zpravidla výchozím shellem na mnoha Unixových systémech, včetně různých Linuxových distribucí, starších verzí macOS[1] a na Windows pomocí Windows Subsystem for Linux. Bash má bohaté skriptovací možnosti, při skriptování je možné využít mnoho funkcionalit, jako například podmínky, cykly, výrazy nebo funkce. Dále Bash umožňuje interaktivní použití, například přijímání vstupu od uživatele, práci se souborovým systémem, správu procesů a mnoho dalšího. (Bertram, 2017)

1.1 Historie a vývoj Bashe

Bash byl v roce 1989 vytvořen Brianem Foxem pro projekt GNU jako svobodná náhrada za Bourne Shell vyvinutý Stephenem Bournem. Cílem bylo vytvořit svobodnou náhradu a zároveň zajistit kompatibilitu s Bourne Shellem a přidat další vlastní funkcionality. Bourne Shell je částečně kompatibilní se standardy POSIX, cílem Bashe bylo zajistit plnou kompatibilitu. Postupné rozšíření umožnilo vytvářet přenositelné skripty, které mohou fungovat na různých systémech s minimálními úpravami.

1.1.1 Projekt GNU

Základní myšlenkou projektu GNU je, že software by měl být svobodný. Vznik projektu byl iniciován Richardem Stallmanem v roce 1983. Součástí projektu je také licence GPL, dnes ve verzi GPLv3. Cílem bylo vytvořit svobodný operační systém podobný proprietárnímu Unixu. V průběhu let vzniklo mnoho nástrojů včetně kompilátoru GCC, shellu Bashe a dalších nástrojů. Na počátku devadesátých let měl projekt množství nástrojů, ale stále chybělo jádro operačního systému. V roce 1991 vydává Linus Torvalds svou první verzi Linuxového jádra. Díky použití licence GPL bylo možné svobodně kombinovat nástroje GNU a jádro Linuxu. Projekt GNU se řídí čtyřmi svobodami. „Program je svobodný software, pokud jeho uživatelé mají čtyři základní svobody: svobodu spouštět program dle vlastního uvážení a jakéhokoliv účelu; svobodu prozkoumávat, jak program funguje a měnit jej, aby vyhovoval vlastním potřebám, přičemž předpokladem je přístup ke zdrojovému kódu; svobodu šířit kopie, abyste pomohli ostatním; svobodu šířit kopie upravených verzí ostatním, díky tomuto může celá komunita těžit z vašich změn, přičemž předpokladem je přístup ke zdrojovému kódu.“ (What is Free Software, c1996-2002, 2004-2019, 2021-2024 - přeloženo autorem).

[1] Od verze macOS 10.15 Catalina je výchozí shell zsh

1.1.2 Verze Bashe

Aktuální stabilní verze je 5.2.[2] Nainstalovanou verzi lze zjistit pomocí příkazu:

```
bash --version
```

Druhou možností je výpis proměnné BASH_VERSION:

```
echo $BASH_VERSION
```

Bash prošel za dobu své existence mnoha verzemi a aktualizacemi. Následují vybrané hlavní verze a jejich vybrané změny:

Verze 1.0 (1994): první oficiální vydání

Verze 2.0 (1996): přidána podpora vlastních pravidel a asociativních polí (Bash changes, 2022)

Verze 3.0 (2004): přidána podpora pro 64bitovou aritmetiku, vylepšena práce s řetězci (Bash changes, 2022)

Verze 4.x (2009–2016): přidána podpora pro zpracování více signálů, přidána podpora pro další architektury, zlepšení stability a opravy zabezpečení (Bash changes, 2022)

Verze 5.x (2019–dosud): vylepšena práce s proměnnými a poli, opravy chyb (Bash changes, 2022)

1.2 Ostatní shelly

Mezi další Unixové shelly patří například zsh, csh, tcsh, ksh, fish. Dostupné shelly jsou evidovány v souboru /etc/shells, tento soubor je možné vypsat pomocí příkazu:

```
cat /etc/shells
```

Zsh (Z shell) je postaven na Bashi. Obsahuje podporu pluginů, doplňování a kontrolu příkazů a má mnohem větší možnost vlastních úprav (Henry-Stocker, 2021).

Csh (C shell) je pojmenovaný po jazyku C a má i podobnou syntaxi. Dnes je méně používaný.

Tcsh (Tee cee shell) je rozšířený C shell. (Henry-Stocker, 2021)

[2] Platné k 8. 2. 2024

Ksh (Korn shell) jehož počátky vzniku jsou před Bashem, je méně uživatelský přívětivý, ale je vhodný pro složitější skripty díky své výkonnosti. (Randal K., 2011)

Fish (Friendly interactive shell) je velice uživatelsky přívětivý. Využívá barevné označení pro konkrétní typy příkazů i pro označení chyb. (Henry-Stocker, 2021)

2 ZÁKLADNÍ PRINCIPY

Stejně jako v Unixových operačních systémech a nástrojích, i v Bashi se rozlišují malá a velká písmena.

2.1 Struktura řádku terminálu

Textový řádek v terminálu se označuje anglickým slovem „prompt“ a je rozdělen na několik částí. Jeho podoba se může lišit dle zvoleného operačního systému nebo shellu. Prompt je možné rozsáhle upravovat. Prompt obvykle začíná uživatelským jménem aktuálního uživatele, následně navazuje název hostitelského počítače oddělený zavináčem. Poté je zobrazena cesta do aktuálního adresáře. Cesta je od názvu hostitelského počítače oddělena dvojtečkou. Posledním zobrazeným znakem bývá znak dolar „$“ nebo znak hash „#“. Znak dolar indikuje, že aktuální uživatel je obyčejný uživatel (bez vyšších práv), zatímco znak hash indikuje superuživatele (uživatele s právy root). Nakonec je prostor na zadávání příkazů.

Obrázek 1: Textový řádek v terminálu

2.2 Interaktivní režim

Při použití interaktivního režimu uživatel zadává příkazy do terminálu a dostává okamžitou odezvu od zadaného příkazu. Tento režim je vhodný pro běžnou práci se systémem, například pro správu systému, pro práci se souborovým systémem nebo pro spouštění aplikací. Odeslání příkazu se provede stisknutím klávesy „Enter“. Pro doplňování příkazu, jeho argumentů nebo parametrů přepínačů je možné využít klávesu „Tab“. V interaktivním režimu je běžně ukládána historie odeslaných příkazů.

Tuto historii lze zobrazit pomocí příkazu:

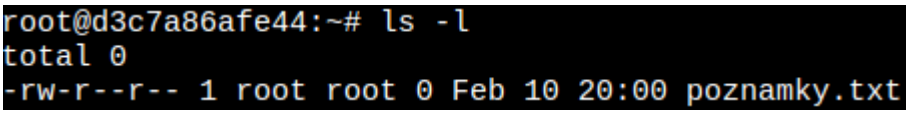
```
history
```

Alternativně lze využít šipky nahoru a dolů.

V historii je možné vyhledávat pomocí příkazu:

```
history | grep [vyhledávaná fráze]
```

Vyhledávat v historii je také možné pomocí klávesové zkratky „Ctrl+R“. Při vyhledávání jednoho slova není potřeba uvádět uvozovky, v případě více slov jsou uvozovky vyžadovány, jelikož mezera nebo tabulátor slouží jako oddělovač jednotlivých příkazů. Při běhu v interaktivním režimu je použita uživatelská konfigurace, která je uložena ve skrytých souborech .bash_profile, .bashrc nebo .profile, které se obvykle nachází v domovském adresáři uživatele.



Obrázek 2: Interaktivní režim

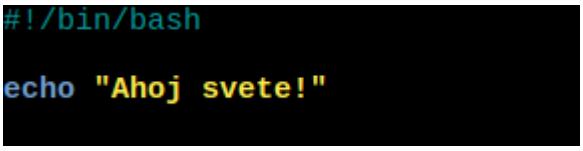
2.3 Skriptovací režim

Tento režim je vhodný pro vytváření skriptů. Pomocí skriptů lze automatizovat běžné, a i díky programovacím možnostem, složitější úkoly. Tento režim může pracovat i interaktivně, kdy skript vyzve uživatele k zadání vstupu. Bash skript lze obvykle identifikovat pomocí přípony „*.sh*“ a takzvaného shebangu:

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

Shebang se skládá ze znaků „#“ a „!“ a z absolutní cesty k interpreteru Bashe. Shebang se nachází na začátku skriptu a informuje systém, že tento skript má být spuštěn pomocí interpreteru Bashe. Přípona .sh není nutná ke spuštění skriptu, ke spuštění je typicky vyžadováno právo spuštění[3] (execute). Díky rozšířenosti je možné vytvářet přenositelné skripty a pro složitější úkoly je možné kód skriptů kombinovat s jinými programovacími jazyky nebo provést integraci s dalšími aplikacemi, například webovými servery, úložišti nebo databázemi. Skripty není potřeba kompilovat, kód skriptů je zpracován interpretem operačního systému. Díky interpretaci bývá obsah skriptů v textově čitelné formě. Čitelnost kódu je užitečná pro validaci kódu, u skriptu z neznámého zdroje je možné zkontrolovat zdrojový kód před samotným spuštěním skriptu.

[3] Není vyžadováno v případě spuštění pomocí příkazu bash



Obrázek 3: Skriptovací režim

3 ZÁKLADNÍ KONCEPTY

Obecná syntaxe příkazu je následující:

```
příkaz [přepínače] [argumenty]
```

Přepínače slouží k ovlivnění chování příkazu, je možné kombinovat více přepínačů zároveň. Argumenty slouží k poskytnutí dodatečných informací příkazu. Mnoho příkazů je možné použít bez jakýkoliv přepínačů či argumentů. I v případě příkazů, přepínačů a argumentů se rozlišují velká a malá písmena.

3.1 Manuálové stránky

Při práci nejen s Bash příkazy a skripty je potřeba vyhledávat bližší informace. Vhodným zdrojem mohou být zabudované manuálové stránky, které jsou obsáhlé, strukturované a bez nutnosti připojení k internetu. Příkaz „man“ slouží k zobrazení manuálových stránek pro příkazy, programy, systémová volání, soubory a mnoho dalších.

Manuálové stránky pro příkaz „man“ lze zobrazit pomocí:

```
man man
```

Manuálové stránky jsou standardně rozděleny do 8 kategorií. Existují i neoficiální kategorie, například pro programy jádra nebo pro hlavičkové soubory jazyka C. (Watson, 2024)

Tabulka 1: Kategorie manuálových stránek

Kategorie	Popis
1	příkazy shellu (uživatelské příkazy)
2	systémová volání (funkce jádra)
3	knihovny
4	speciální soubory (zpravidla ve složce /dev)
5	formáty konfiguračních souborů (například /etc/passwd)
6	hry

7	různé (například man-pages(7))
8	příkazy pro administraci systému (typicky pro uživatele root)

Konkrétní kategorii manuálové stránky lze specifikovat v příkazu. V případě příkazu man lze kategorii „různé“ otevřít pomocí příkazu:

```
man 7 man
```

Struktura manuálové stránky by měla obsahovat minimálně tyto části:

Tabulka 2: Sekce manuálové stránky

Část	Popis
NAME	název
SYNOPSIS	shrnutí syntaxe příkazu
DESCRIPTION	popis
SEE ALSO	další související manuálové stránky

Manuálová stránka může obsahovat i doplňující části, některé pouze u vybraných kategorií, například konfigurace, návratové hodnoty, chyby, verze, atributy, chyby, příklady, autoři. (Kerrisk, 2020)

3.1.1 Lokalizované stránky

Běžně se manuálové stránky zobrazují ve výchozím jazyce, v případě, že není manuálová stránka lokalizována do specifického jazyka, zobrazí se stránka v angličtině.

Tímto příkazem je možné vynutit zobrazení manuálové stránky pro příkaz man v češtině[4]:

```
man -Lcs man
```

[4] Vyžadovanou lokalizaci je nutné mít nainstalovanou předem

3.1.2 Vyhledání manuálové stránky

Pokud nelze nalézt manuálovou stránku, je vhodné využít příkaz man s přepínačem „-k“. (Watson, 2024)

Pomocí následujícího příkazu je zobrazen výpis všech manuálových stránek souvisejících s kopírováním:

```
man -k copy
```

3.2 Základní příkazy

Příkaz echo

Tento příkaz vypíše řádek textu do standardního výstupu. Výchozí volba je ignorovat speciální znaky za zpětným lomítkem takzvané „backslash escapes“. Pomocí přepínače „-e“ je možné zapnout interpretaci těchto speciálních znaků. Použití tohoto příkazu je vhodné zejména ve skriptech, kdy je potřeba informovat uživatele. (Fox, c1994–2024)

Příkazy whoami, id

Tyto příkazy slouží pro přehled uživatelů, uživatelských a skupinových ID.

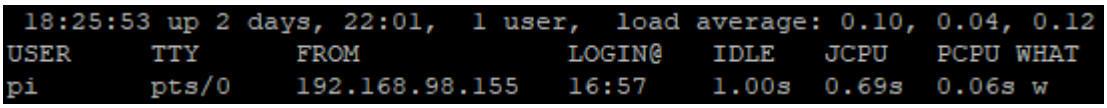
Příkazem „whoami“ lze získat uživatelské jméno efektivního uživatele[5]. Příkaz slouží k zobrazení reálných a efektivních uživatelských a skupinových informací. Pokud není specifikován uživatel, jsou vypsány informace o aktuálním uživateli. Pomocí přepínačů lze upravit výstup příkazu. Přepínače „-u“ a „-g“ slouží k výpisu pouze efektivních uživatelských, respektive skupinových ID. (Robbins, c1994–2024), (Mlynarik, c1994–2024)

Příkaz „id“ včetně jeho přepínačů je vhodný zejména pro kontrolu správného členství uživatele ve skupinách.

Příkaz w

Příkaz zobrazuje informaci, jací uživatelé jsou přihlášeni, název jejich terminálu, IP adresu pouze pokud je uživatel přihlášen vzdáleně, čas přihlášení, dobu neaktivity, využití procesorového času procesy konkrétního terminálu (JCPU), využití procesorového času aktuálním příkazem (PCPU) a příkaz, který aktuálně probíhá v terminálu. (Blake, c1994–2024)

[5] Identita uživatele, pod kterou probíhá proces



Obrázek 4: Výstup příkazu „w“

3.3 Příkazy souborového systému

Příkaz pwd

Tento příkaz zobrazí absolutní cestu aktuálního pracovního adresáře. Zkratka PWD je tvořena z anglických slov Print Working Directory, což v překladu do českého jazyka znamená „*vypiš cestu k pracovnímu adresáři*“. Ve výchozím nastavení příkaz nevypisuje cesty symbolických odkazů. (Meyering, c1994–2024)

Příkaz cd

Příkaz slouží ke změně adresáře. Jedna tečka reprezentuje aktuální adresář, dvě tečky reprezentují nadřazený adresář.

Cestu je možné zadávat absolutně i relativně. Vhodné je i využití znaku lomeno „/“, pro přesun do kořenového adresáře nebo také znaku vlnovka „~” pro přesun do domovského adresáře uživatele, pro přesun do domovského adresáře lze také použít samotný příkaz „cd“ bez jakýkoliv argumentů.

Příklad změny adresáře pomocí absolutní a následně relativní cesty, v obou případech je aktuální pracovní adresář domovský uživatele Karel:

cd /home/Karel/data/2018
cd data/2018

Příkaz ls

Tento příkaz vypíše obsah složky. Obvykle se využívá společně s přepínačem „-l“, který vypíše více informací přehledněji. Přepínač „-a“ slouží k vypsání obsahu složky včetně skrytých souborů. Skryté soubory jsou identifikovatelné pomocí názvu začínajícího tečkou. (Stallman, c1994-2024)

Příkazy mkdir, rmdir

Příkaz „mkdir“ slouží k vytváření složek. Pro vytvoření složky stačí zadat za příkaz zadat název složky, pokud složka již existuje, je zobrazena se chybová hláška. S využitím přepínače „-p“ je

možné vytvářet přímo i podsložky, a pak už existence jakékoliv složky se stejným názvem nebrání úspěšnému provedení příkazu. (MacKenzie, 2024)

Příkaz „rmdir“ lze naopak odstranit prázdných složek. Syntaxe příkazu je stejná jako u příkazu „mkdir“. Obsahuje také přepínač „-p“ i pro odstranění podsložek. Pro odstranění složek je také možné využít následující příkaz „rm“.

Příkazy touch, rm

Příkaz „touch“ slouží primárně k změně časových razítek u souboru. Vedlejší funkcí tohoto příkazu je vytvoření prázdného souboru. Pokud není potřeba vytvořit soubor, je možné využít přepínač „-c“ či jeho delší variantu „--no-create“. Pomocí přepínače „-a“ lze vynutit pouze změnu časového razítka přístupu do souboru, pomocí přepínače „-m“ lze vynutit pouze změnu časového razítka úpravy souboru. (Rubin, c1994–2024)

Pomocí příkazu „rm“ je možné odstranit soubory nebo složky. Syntaxe je stejná jako u předchozího příkazu. (Rubin, c1994–2024)

Příkazy cp, mv

Pomocí příkazu „cp“ je možné kopírovat soubory a složky. Je také možné zároveň zkopírovat více souborů do jedné složky. (Granlund, c1994–2024)

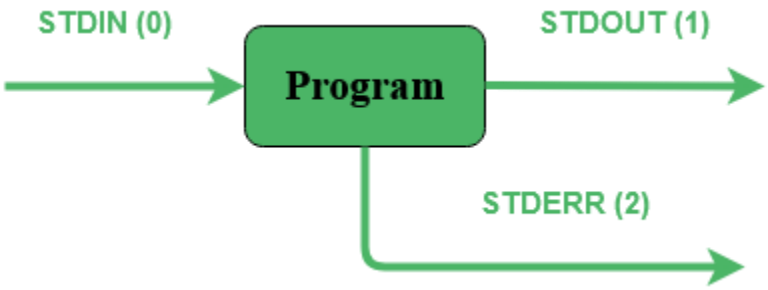
Syntaxe příkazu je následující:

```
cp [přepínače] [zdroj/zdroje] [cíl]
```

Příkaz „mv“ má dvě funkce. Slouží k přesunutí souborů, složek a zároveň také slouží k přejmenování souborů a složek. Příkaz obsahuje méně přepínačů než příkaz „cp“, chybí přepínače na pevné a symbolické odkazy a na přesunutí rekurzivně. (Parker, c1994–2024), (Kameník, 2011)

3.4 Vstupy, výstupy

Standardně má každý unixový program 3 proudy, pro vstup, pro výstup a pro chybový výstup. (Stdin, stdout, stderr - standard I/O streams, 2017)



Obrázek 5: Vstup a výstupy programu (Rossi, 2019)

3.4.1 Standardní vstup

Standardní vstup je označován jako „stdin“. Další označení je „<“ nebo také „0<“. Vstupem může být vstup z klávesnice nebo ze souboru. Pro vstup ze souboru lze použít levou šipku „<“. (Holt, 2018), (Whittal, 2005)

3.4.2 Standardní výstup

Standardní výstup je označován jako „stdout“, také jako „>“ nebo „1>“. Výstup je typicky zobrazen v uživatelově terminálu, může být i přesměrován do souboru, vždy záleží odkud proces pochází.

Následujícím příkazem lze přesměrovat výstup výpisu složky do souboru s názvem „export“:

```
ls -lh > export
```

Pokud soubor neexistuje, bude vytvořen, pokud již existuje bude přepsán aktuálním obsahem. Pro zachování stávajícího obsahu souboru a přidání nového vstupu je nutné použít dvakrát znak přesměrování „>>“. (Holt, 2018), (Whittal, 2005), (Ramey, 1998)

3.4.3 Standardní chybový výstup

Posledním výstupem je chybový výstup, označovaný jako „stderr“ nebo jako „2>“. Tento výstup poskytuje informace o chybách při zpracování příkazu. Chybový výstup je typicky zobrazen v uživatelově terminálu, stejně jako v případě standardního výstupu. Je také možné ho přesměrovat do souboru, což je vhodné pro pozdější analýzu chyb při běhu programu. (Holt, 2018), (Whittal, 2005), (Ramey, 1998)

Přesměrování výstupů je možné kombinovat, standardní výstup bude přesměrován do souboru s názvem „export“ a chybový výstup bude přidán do souboru s názvem „zaznamChyb“:

```
ls -lh 1> export 2>> zaznamChyb
```

Pro skrytí chybového výstupu v terminálu je možné použít přesměrování do takzvané černé díry, což je soubor s absolutní cestou „/dev/null“.

3.5 Roury

Roura propojuje výstup příkazu se vstupem jiného příkazu. Propojení se provádí použitím znaku roura „|”. Lze použít například příkaz „ls“ pro výpis obsahu složky a příkaz „grep“ pro filtrování výpisu obsahu.

Roury je možné kombinovat, pomocí příkazu „wc -l“ získán počet řádků.

Příkladem může být výstup následujících příkazů, který zobrazí počet souborů obsahující slovo „bin“:

```
ls -lh | grep bin | wc -l
```

Pojmenované roury

Pro uložení obsahu do roury a následné využití je možné použít pojmenovanou rouru. Pojmenované roury pracují jako datová struktura FIFO. Pojmenovanou rouru lze vytvořit příkazem „mkfifo“ následovanou názvem roury. Rouru je možné naplnit například přesměrováním výstupu příkazu výpisu obsahu složky a obsah roury zpracovat například z jiného terminálu, například příkazem „cat“ je možné vypsat obsah pojmenované roury. Po použití je roura prázdná a je do ní možné uložit další obsah. Do roury je možné i přidávat obsah pomocí dvou pravých šipek „>>“. Rouru lze odstranit jako běžný soubor, příkazem „rm“. (Vaught, 1997), (Wang, 2018)

3.6 Příkazy pro práci s textem

Příkaz grep

Tento příkaz slouží pro vyhledávání textového výskytu v textu nebo souborech. Vyhledávat je možné i pomocí regulárních výrazů. Mezi další funkce patří například spočtení výskytů nebo procházení mnoha souborů v jednom příkazu. Příkaz je velice oblíbený díky své efektivitě a rychlosti, a to i při velkých datových sadách. (Free Software Foundation, Inc., 2019)

Příkaz awk

Tento příkaz vychází z programovacího jazyka „AWK“. Typicky je používán pro zpracování textových dat. Možnosti zpracování dat jsou rozsáhlé, je možné využít pravidla, vzory a další.

Je možné pracovat jak s textovými, tak numerickými či časovými daty. Použití může být například pro načtení obsahu souboru, následné vyfiltrování a vypsání výsledku do terminálu. Příkaz může být alternativně použit i pro vypsání celého souboru. (The GNU Awk User's Guide, 2020)

Příkaz sed

Příkaz „sed" slouží k výpisu, přidávání, nalezení, úpravám a odstranění textu. Obvykle bývá používán k úpravám textových souborů. Na rozdíl od příkazu „awk" je „sed" přímo určen pro úpravu textu, díky tomu má jednodušší syntaxi, ale zase omezenější funkce. Příkazem je možné nahrazovat konkrétní výskyty slov, doplňovat nová slova nebo odstraňovat celé řádky. (Meyering, 2020)

Příkaz cmp

Příkaz slouží k porovnání dvou souborů po bajtech. Díky tomu nemusí být porovnávané soubory v textově čitelné podobě, může se jednat o zkompilované programy nebo třeba obrázky. Při výpisu příkaz zobrazí, na kterých bajtových pozicích se soubory liší. Při porovnávání je možné upravit rozsah porovnání, například přeskokem na konkrétní bajt nebo kontrolou pouze konkrétního počtu bajtů. (Meyering, 2023)

Příkaz diff

Název příkazu vychází z anglického slova „difference", což znamená v češtině rozdíl. Příkaz slouží k porovnání více souborů po řádcích. Na rozdíl od příkazu „cmp", který porovnává po bajtech, tento příkaz srovnává celé textové řádky. Výstupem příkazu jsou pouze rozdílné řádky. Příkaz je vhodný pro kontrolu změn v různých verzích souboru. (Meyering, 2023)

4 POKROČILÉ KONCEPTY

Tato kapitola se věnuje vytváření Bash skriptů a představení vybraných struktur včetně rozdílů oproti jiným vyšším programovacím jazykům. V Bash skriptech lze nalézt mnoho struktur od proměnných, podmínek, smyček až po funkce.

Bash skripty jsou z většiny psány v obyčejné textové podobě a začínají takzvaným shebangem „#!/bin/bash“. Pro spuštění musí mít uživatel povoleno právo spuštění (execute), které lze nastavit pomocí příkazu „chmod +x bashSkript.sh“ nebo spustit skript přímo pomocí příkazu „bash“. Skripty jsou typicky označeny příponou „.sh“, ale její přítomnost není podmínkou ke spuštění skriptu. Následně je možné skript spustit zadáním cesty ke skriptu v případě terminálu, v případě grafického rozhraní zpravidla dvojitým stiskem levého tlačítka myši.

4.1 Prostředí pro psaní Bash kódu

Pro Bash nejsou standardně dostupná plnohodnotná IDE pro psaní a spouštění kódu jako u vyšších programovacích jazyků (Java, C#, Python apod.). Pro zvýšení efektivity a přehlednosti kódu je možné využít rozšíření do existujících IDE nebo zvolit pokročilé textové editory. Tato prostředí či editory zpravidla disponují vybranými funkcemi jako například zvýraznění zdrojového kódu, detekcí chyb, návrhy a doplňování zdrojového kódu, verzování změn, správou souborů a nástroji pro ladění programu.

Následují vybraná IDE pro Bash a jejich krátký popis:

- Visual Studio Code (podpora přes rozšíření např. Bash Debug, Bash IDE, shell-format),
- Eclipse (primárně určeno pro Javu, podpora pro Bash pomocí rozšíření „ShellWax“)
- IntelliJ IDEA (primárně určeno pro jazyky využívající JVM, podpora pro Bash pomocí rozšíření „BashSupport Pro“). (Jain, 2022), (JetBrains s.r.o., c2000-2024), (Eclipse Foundation AISBL, 2024)

Vybrané textové editory s podporou pro Bash kód a jejich krátký popis:

- Kate (textový editor grafického prostředí KDE Plasma),
- Notepad++ (textový editor pro Windows),
- Vim (vylepšená verze editoru „Vi“),
- Sublime Text (multiplatformní textový editor s podporou značkovacích jazyků).

4.2 Proměnné

Proměnné v Bashi nejsou omezeny specifickým datovým typem jako v jiných vyšších programovacích jazycích jako například Java, C# nebo C. Veškerá data jsou v proměnných primárně ukládána jako textové řetězce[6]. Proměnné jsou dynamicky typované, v případě součtu dvou číselných hodnot ve dvou různých proměnných je provedena automatická konverze, není tedy nutné ruční přetypování na číselný datový typ. Do proměnných lze uložit čísla, textové řetězce nebo i například pole. Bash nevyžaduje u proměnných deklaraci, proměnné jsou rovnou definovány, stejně tak se neprovádí ruční zrušení proměnné. (Maleki, 2022), (Herbst, 2017)

Následující příkaz vytvoří novou proměnou s názvem „jmeno“ a do ní uloží hodnotu „Karel“:

```
jmeno=“Karel“
```

K hodnotám proměnných se přistupuje pomocí znaku dolar „$“, takto lze obsah proměnné vypsat:

```
echo $jmeno
```

Bash podporuje i lokální proměnné, využívají se v uzavřených blocích kódu. Lokální proměnnou lze vytvořit pomocí klíčového slova „local“ před názvem proměnné.

Stejně jako názvy souborů i u proměnných jsou rozlišována velká a malá písmena. Některé názvy proměnných nesmí být použity. Jedná se o vybraná rezervovaná slova, například if, for, export a další. Název proměnné také nesmí začínat číslem. Při pojmenování proměnných je vhodné dodržovat konvence názvů proměnných. Konstanty a proměnné z proměnného prostředí jsou typicky uváděny velkými písmeny, zatímco běžné proměnné jsou uváděny malými písmeny. Při použití víceslovných proměnných je vhodné oddělovat slova podtržítkem (snake_case). Platnost proměnné je dána tím, ve kterém kontextu byla proměnná vytvořena. Při definici proměnné v Bash skriptu je platnost pouze v rámci skriptu, při definici v terminálu je proměnná platná do ukončení terminálu, typicky do odhlášení uživatele. (Billemont, 2023)

4.3 Pole

Pole umožňují ukládat více textových řetězců do jednoho celku. V Bash existují dva druhy polí. Běžná indexovaná pole a od verze Bash 4 i asociativní pole. Podobné datové struktury jako

[6] Lze změnit pomocí příkazu “declare”

asociativní pole v jiných programovacích jazycích představují například mapy nebo slovníky. Ukládání dat v asociativních polích funguje na principu klíč-hodnota. Veškerá pole jsou dynamická, není je tedy nutné předem deklarovat na předem stanovenou velikost. Nevýhodou dynamických polí je menší výkon oproti statickým polím. U Bash skriptů to není tak významný problém, protože jsou určeny pro automatizaci a jednodušší práci s daty. Stejně jako proměnné, pole nejsou omezena specifickým datovým typem. Je tedy možné v jediném poli kombinovat textové řetězce a čísla. (Billemont, 2023)

Indexované pole je možné vytvořit například následovně:

```
rostliny=(„Monstera“ „Fíkus“ „Kaktus“)
```

Pro přístup ke konkrétnímu indexu či přidání nové položky pole postačuje zapsat index do hranatých závorek, například rostliny[3]=“Olivovník“.

Na rozdíl od indexovaných polí, asociativní pole je nutné deklarovat před použitím.

Asociativní pole je možné deklarovat následovně:

```
declare -A asociativniPole
```

Přístup či přidávání prvků v případě asociativních polí funguje obdobně. Namísto číselné hodnoty indexu je do hranatých závorek zapsán identifikátor klíče, zpravidla jakýkoliv textový řetězec. Odstranění prvku z pole lze provést přidáním klíčového slova „unset“ před název pole s indexem nebo klíčem. Příkazem „unset“ je možné odstranit i celé pole, za příkaz stačí dosadit název pole bez jakéhokoliv indexu nebo klíče. Při dosazení znaku zavináč „@“ namísto indexu nebo klíče lze přistupovat ke všem prvkům zároveň. Použití tohoto znaku je vhodný například při odstranění všech prvků pole, při procházení pole nebo pro zjištění velikosti pole. Pro výpis velikosti pole je nutné přidat před název pole znak hash „#“ a samotné vypsání číselné velikosti lze provést například příkazem „echo ${#rostliny[@]}”.

Pro výpis obsahu pole je možné využít nějaký cyklus jako „for“, „while“ nebo je opět možné využít znak zavináč v indexu nebo klíči pole, který zajistí výpis všech položek pole. (Ramey, 1998)

4.4 Řízení toku

Stejně jako v jiných programovacích jazycích, v Bash skriptech je možné ovlivnit vykonávání částí programu pomocí příkazů pro řízení toku programu. Mohou to být podmínky, které vykonají část kódu programu při splnění nebo nesplnění podmínky (zpravidla Booleovského typu) nebo cykly, které umožňují opakované vykonání stejné části kódu programu.

4.4.1 Podmínky

Z podmínek jsou podporovány: podmínky „if“, „if-else“, „if-elif-else“ a podmínka „case“. Logická podmínka, která má být vyhodnocena, je zapisována za klíčovým slovem „if“ do hranatých závorek, tato závorka je ukončena středníkem. Po podmínce následuje klíčové slovo „then“ a kód, který má být proveden. Následně je možné podmínkovou strukturu ukončit slovem „fi“ nebo rozšířit o další podmínky pomocí „elif“. (Herbst, 2017)

4.4.2 Cykly

Bash obsahuje běžné známé cykly „for“, „while“ a navíc obsahuje méně častý cyklus „until“, což je v podstatě cyklus „while“, kde je podmínka vyhodnocována opačně. Bash neobsahuje cyklus typu „do-while“, kde podmínka zajistí minimálně jednou vykonání kódu cyklu, následně se vyhodnocuje podmínka, která rozhodne o dalších opakováních. (Herbst, 2017)

4.5 Funkce

Funkce umožňují zapouzdřit více různých příkazů do jednoho uceleného bloku, který je možné opakovaně volat pomocí jeho názvu. Funkce pomáhají lepší organizaci kódu, omezují duplicitní kód, díky tomu je zlepšena čitelnost kódu a pozdější údržba. Definice funkce může být provedena dvěma způsoby. Jedna možnost je rovnou zapsat název funkce a za ní dvojici kulatých a složených závorek, ve složených závorkách bude kód. Tento způsob je kompatibilní se standardem POSIX, tento kód bude možné spustit větším množství různých operačních systémů. Druhá možnost, také někdy nazývána jako „Bash varianta“, přidává klíčové slovo „function“ a vynechává dvojici kulatých závorek. Zjevným rozdílem od jiných vyšších programovacích jazyků je práce s parametry funkcí. Datové typy a počty parametrů nejsou uváděny u definice funkce. Ve funkci je možné se odkazovat na parametry pomocí znaku dolar „$“ a čísla. Pomocí speciální kombinace znaků dolar a znaku zavináč je možné se odkazovat na veškeré parametry zároveň, parametry jsou uloženy do jediného pole. (Both, 2020)

5 IMPLEMENTACE

Tato kapitola se zabývá implementací webové aplikace pro výuku jazyka Bash. Nejprve jsou představeny požadavky, následně použité softwarové nástroje a poté samotné rozhraní a funkce webové aplikace.

5.1 Požadavky

Cílem bylo vytvořit webovou aplikaci pro výuku skriptovacího jazyka Bash. Požadavky na aplikaci byly následující:

- Výukový obsah rozdělen do kapitol obsahující následující témata:
  - Základní práce s příkazovým řádkem,
  - Práce se souborovým systémem,
  - Manipulace s textem (výstupy, zabezpečení vstupu),
  - Proměnné, datové typy,
  - Podmínky, cykly,
  - Pole, funkce,
  - Správa procesů,
  - Plánování (zautomatizování) úloh,
  - Tipy pro práci s Bashem,
- Přístup k terminálu se shellem Bash s možností zkoušet příkazy a skripty,
- Nezávislý prostor terminálu, kde nebudou uživatelé vzájemně ovlivňováni s možností rychlého obnovení do čistého stavu v případě jakýkoliv chyb,
- Skripty sloužící k prověření znalostí,
- Sekce pro správce a editory, dostupná po přihlášení s možností přidávání, upravování a odstraňování jednotlivých kapitol, učebních textů a správou uživatelů.

5.2 Existující aplikace

Na internetu je dostupných mnoho textů o skriptování v Bash. Jen minimum z nich poskytuje přímý přístup k terminálu se shellem Bash a tím možnost rovnou zkoušet a získávat zkušenosti od plnohodnotného Bash shellu. V českém jazyce jsou dostupné kurzy, ale žádný běžně dostupný kurz neposkytuje přístup k terminálu. V následujícím textu jsou představeny tři kurzy, z toho dva v anglickém jazyce a jeden v českém.

5.2.1 Codeacademy – Learn Bash Scripting

Codeacademy nabízí interaktivní webový kurz. Po rychlé registraci je dostupný krátký kurz obsahující sedm částí. Kurz slouží pouze pro seznámení se se skriptováním. Kurz začíná základy vytvoření skriptu, následně seznámí účastníka kurzu s proměnnými a dalšími strukturami jako jsou podmínky, cykly a funkce. Celý kurz je veden v anglickém jazyce a webová stránka má také podporu pro mobilní zařízení.

5.2.2 Datacamp – Introduction to Bash Scripting

Kurz od společnosti Datacamp je mnohem obsáhlejší. Kurz je spíše určen pro uživatele, kteří již mají alespoň základní znalosti příkazového řádku a chtějí si je rozšířit o znalosti skriptování a automatizace. Po registraci je volně dostupná pouze část kurzu obsahující základní příkazy a základy skriptů, pro další části je nutné zakoupit měsíční předplatné.

5.2.3 ITnetwork – Skriptování v Bash

Česká firma ITnetwork poskytuje pro výuku skriptování v Bash online webový kurz. Kurz je volně dostupný na internetu, bez nutnosti registrace. Volný přístup je umožněn k prvním třem lekcím kurzu. V těchto lekcích se uživatel dozví o teoretických základech Bash, proměnných a na závěr o uživatelských vstupech do skriptu. Zbytek kurzu je placený. Kurz neumožňuje přístup k terminálu přes webový prohlížeč, je nutné využít nějakou alternativu, ať už terminál na vlastním zařízení nebo například službu poskytující online terminál.

5.3 Použité nástroje

5.3.1 Node.js

Node.js je multiplatformní prostředí JavaScriptu s otevřeným zdrojovým kódem, které bylo představeno v roce 2009 Ryanem Dahlem. Prostředí je navrženo na psaní kódu na straně serveru i klienta. Aplikace Node.js probíhá pod jedním procesem, bez nutnosti vytvářet vlákno pro každý požadavek. Cílem autora při vytváření prostředí bylo omezit blokování procesů a tím umožnit vyšší počet současných spojení. Díky tomu umožňuje Node.js vytvářet škálovatelné aplikace s vysokou propustností dat. Node.js používá existující jazyk JavaScript. (OpenJS Foundation, 2024)

Node.js obsahuje balíčkovacího správce s názvem „NPM“. Zkratka „NPM“ znamená anglicky Node Package Manager. Tento balíčkovací správce slouží k instalaci dodatečných knihoven a jejich závislostí. Mezi další funkcionality patří například správa konkrétních verzí balíčků nebo skenování známých bezpečnostních zranitelností. Stejně jako Node.js, NPM má otevřený zdrojový kód. Informace o potřebných balíčcích a jejich závislostech vybraného projektu jsou

uloženy v souboru s názvem „package.json“. Tento soubor může obsahovat i rozdílné verze balíčku pro vývoj nebo skripty, které mohou zautomatizovat každodenní úkoly. Pokud soubor neexistuje, je možné ho vygenerovat pomocí příkazu „npm init“. Pro instalaci balíčku je možné použít příkaz „npm install <název balíčku>“. (Karrys, 2023), (Satheesh, 2015)

5.3.2 JavaScript

JavaScript je multiplatformní vyšší interpretovaný programovací jazyk vytvořen v roce 1995 Brendanem Eichem. JavaScript nijak nevychází z programovacího jazyka Java, pouze má podobný název z důvodu popularity programovacího jazyka Java. Tento programovací jazyk umožnil vytvářet webové stránky s interaktivními prvky. Primárně byl určen pro běh na straně klienta, nejčastěji webových stránek, ale například v Node.js je ho možné používat i pro spouštění kódu na straně serveru. JavaScript se nepoužívá nejen pro webové stránky, ale i pro běžné desktopové aplikace. Pro JavaScript existuje mnoho rozšíření jako třeba celé frameworky nebo knihovny. Mezi nejznámější JavaScriptové frameworky patří React, Angular nebo Vue.js. JavaScript je podporován všemi moderními webovými prohlížeči, je ho tedy možné spustit na téměř jakémkoli zařízení bez ohledu na operační systém. Společně s JavaScriptem vznikl i formát JSON. Na JavaScriptu vznikl také kompilovaných jazyk TypeScript, který se snaží řešit některé nevýhody JavaScriptu. (Minnick, 2023)

5.3.2.1 JSON

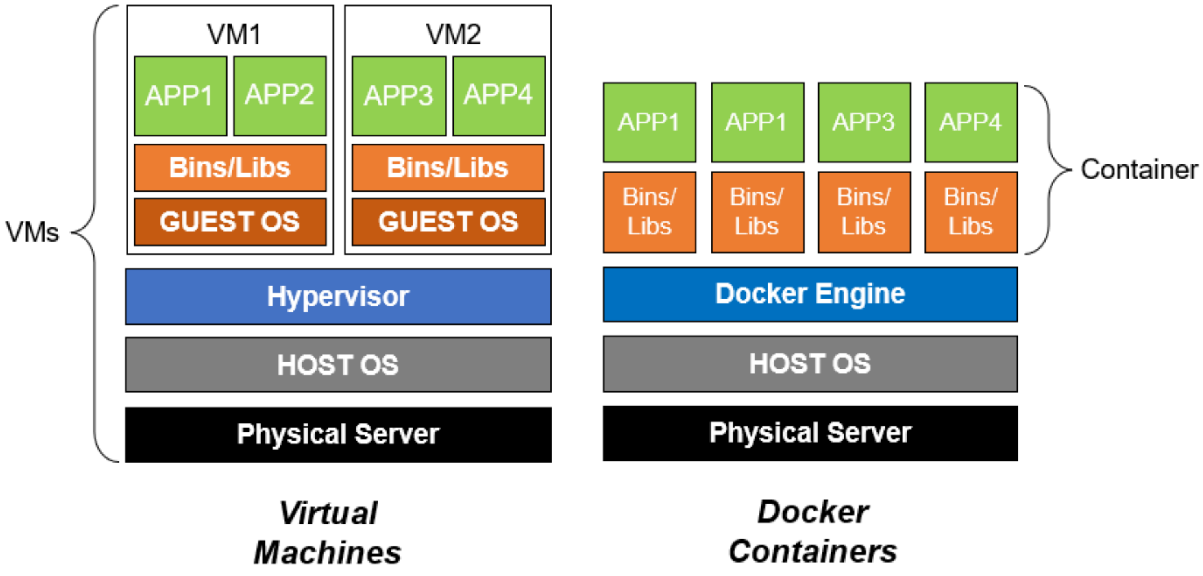
Zkratka JSON znamená JavaScript Object Notation. Jedná se o univerzální formát pro přenos strukturovaných dat mezi různými klienty nebo třeba programovacími jazyky. Vyniká svojí rychlostí i při velkém množství dat. Díky strukturovanému zápisu je snadno čitelný i pro lidi. Velmi často je JSON používán pro API, používán také bývá pro konfigurační soubory aplikací. Díky velkému rozšíření existuje podpora pro práci s JSON v téměř všech moderních programovacích jazycích. Objekty se v JSON zapisují jako dvojice klíč a hodnota, celý objekt je pak obalen složenými závorkami. Před JSON formátem byl velmi rozšířený formát XML, který je dnes na ústupu, ale stále používán ve starších aplikacích. Existuje i binární varianta, označovaná jako BSON. (Smith, 2015)

5.3.3 Docker

Docker je softwarová platforma umožňující vytváření, testování a spouštění aplikací v nezávislých kontejnerech. Poprvé byla představena v roce 2013 firmou dotCloud, Inc., později přejmenovanou na Docker, Inc. Je naprogramována v multiparadigmatickém programovacím jazyku Go a distribuována pod licencí Apache 2.0. Pomocí kontejnerů je

možné vytvořit balíček aplikací včetně všech závislostí. Tento balíček je následně přenositelný a spustitelný v jakémkoliv prostředí bez rozdílu v nastavení. Díky tomuto řešení je zajištěna stejná funkčnost na běžném uživatelském počítači, tak i po přesunu pro reálné nasazení například v cloudu. (Iliev, 2021)

Docker nevirtualizuje celý operační systém, ale využívá pouze virtualizaci na úrovni jádra operačního systému. To znamená, že kontejnery sdílí hostitelský operační systém a obsahují pouze vlastní aplikaci a její závislosti. Podmínkou je kompatibilita jádra hostujícího a virtualizovaného systému. Díky tomu mají kontejnery mnohem nižší nároky na systémové zdroje, což umožňuje na stejném hardwaru provozovat více aplikací. Tato vlastnost také umožňuje rychleji spouštět nové kontejnery a odpadá potřeba správy více celých operačních systémů. Mezi nevýhody patří menší izolace než u běžné virtualizace, chyba v jádře může ovlivnit veškeré kontejnery. Na omezení lze také narazit u aplikací, které vyžadují přímý přístup k hardwarovým zařízením. (Srivastav, 2023)



Obrázek 6: Rozdíl mezi tradiční a kontejnerovou virtualizací (Kim, 2022)

Docker využívá mnoho funkcí Linuxového jádra, například pro izolaci jmenné prostory (namespaces) nebo kontrolní skupiny (cgroups) pro správu zdrojů kontejneru. Jako souborový systém využívá Docker Union File System označovaný zkraceně jako UnionFS. Tento souborový systému udržuje adresáře a data ve vrstvách, následně se celý souborový systém jeví jako jeden ucelený souborový systém. Při spuštění kontejneru se pouze přidá na vrchol stávajících vrstev nová zapisovatelná vrstva. Díky tomu mohou kontejnery sdílet stejné části

a aplikovat změny až na vyšších vrstvách, což umožňuje šetřit diskový prostor a zrychlit spuštění kontejnerů.

Docker se skládá z několika částí:

- Docker Engine
    - Docker démon: Proces na hostitelském operačním systému, spravuje kontejnery, svazky, sítě. Umožňuje komunikaci mezi dalšími démony,
    - REST API: Zajišťuje komunikaci mezi Docker klienty a démony,
    - Docker CLI: Příkazová řádka pro správu kontejnerů,
    - Docker Images: Šablony obrazů určené k vytváření kontejnerů,
    - Docker Containers: Prostředí pro správu a běh kontejnerů,
    - Docker Networks: Prostředí pro správu běh sítí,
    - Docker Volumes: Mechanismus pro persistentní ukládání dat kontejnerů.

5.3.3.1 Docker compose

Docker compose je nástroj, který slouží k provozu více kontejnerů zároveň. Konfigurace se ukládá do strukturovaného souboru zpravidla s názvem „compose.yaml“ nebo „docker-compose.yaml“. Následně je možné pomocí příkazu „docker-compose“ spouštět, zastavovat a spravovat všechny kontejnery vybraného konfiguračního souboru. (Docker Inc., c2013-2024)

5.3.4 SQLite

SQLite je knihovna, která implementuje SQL relační databázový systém. Poprvé byla vydána v roce 2000 Dwaynem Richardem Hippem. Knihovna je napsána v jazyce C a je dostupná pod licencí „public domain“, která umožňuje volný přístup komukoliv pro jakýkoliv účel. Velikost celé knihovny se pohybuje okolo 750 KiB[7]. V knihovně SQLite je implementován téměř celý standard SQL včetně pokročilých funkcí jako například částečné indexy, podpora formátu JSON nebo okenní funkce. (Features Of SQLite, 2023)

SQLite nevyžaduje vlastní server pro provoz databáze. Celá databáze je uložena do jednoho souboru na jakémkoliv běžném úložišti. Díky této skutečnosti nevyžaduje žádnou administraci a konfiguraci serveru. Předností ukládání do jednoho souboru je možnost přenositelnosti mezi různými architekturami nebo operačními systémy. Nevýhodou tohoto řešení je omezená škálovatelnost a při větší rozsáhlosti databáze fragmentace a množství přístupů. Proto je vhodná

[7] S drobnými rozdíly dle platformy a optimalizace kompilátoru

pro menší až střední aplikace nebo jako alternativa k nestrukturovanému ukládání dat do běžných textových souborů či jiných proprietárních formátů.

Pro vytvoření a úpravu databáze je možné využít shell prostředí nebo aplikaci s grafickým prostředím, například „DB Browser for SQLite“.

5.3.5 Xterm.js

Xterm.js je knihovna s otevřeným zdrojovým kódem, která slouží k integraci terminálu do webových aplikací. Tuto knihovnu lze použít i v Electron aplikacích. Mezi podporované procesy patří nejen Unixové shelly jako bash, zsh, tcsh, ale i nástroje obsahující příkazovou řádku nebo shelly programovacích prostředí. Knihovna sama o sobě neposkytuje terminál, umožňuje propojení s procesy terminálů. Xterm.js je používán i ve známých aplikacích jako například Microsoft Visual Studio Code nebo ve virtualizační platformě Proxmox VE. Knihovna se vyznačuje rozsáhlou kompatibilitou, podporovány jsou všechny moderní prohlížeče, bez nutnosti instalaci jakýkoliv dalších komponent či aplikací. Při vývoji je zohledňována co nejnižší doba odezvy a vysoká propustnost dat. Pro obousměrnou komunikaci v reálném čase je používán protokol Websocket. (SourceLair, 2024)

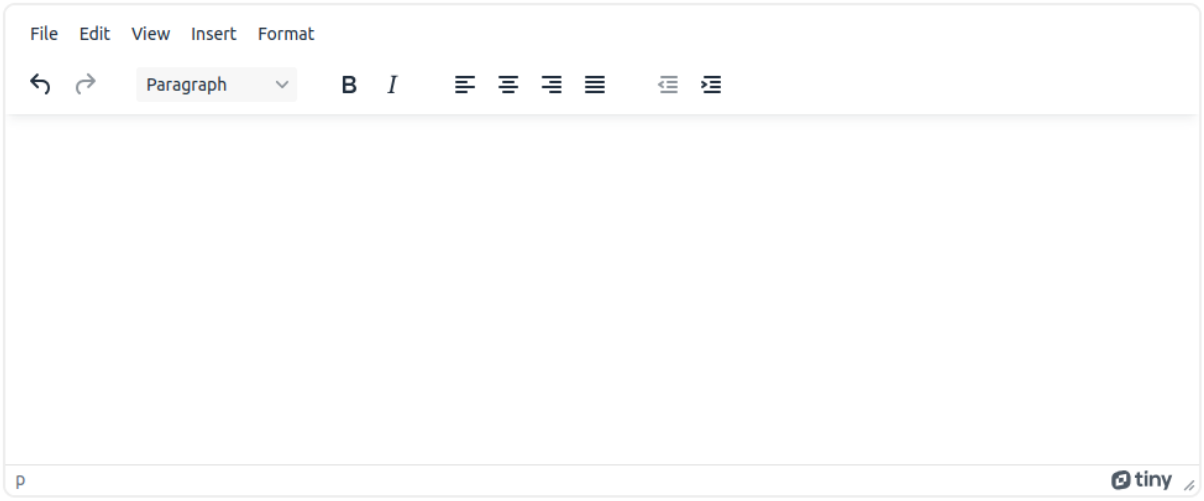
5.3.6 Protokol Websocket

WebSocket je protokol, který umožňuje obousměrnou komunikaci klient-server v reálném čase pomocí jediného TCP spojení. Poprvé byl představen v roce 2008 a následně v roce 2011 standardizován organizací IETF. Na rozdíl od běžného http modelu „požadavek a odpověď“, WebSocket nevyžaduje otevření spojení pro každý požadavek. Pro spojení využívá běžný port 443, případně port 80, které obvykle nejsou filtrovány, je ale možné použít libovolný port. Spojení může být uzavřeno buď klientem nebo serverem, jakmile klient nebo server uzavře spojení, uzavře se spojení i na druhé straně. Použití není omezeno pouze na webové aplikace, WebSocket může být využit i u mobilních aplikací, desktopových aplikací nebo při komunikaci s IoT senzory. Použití je vhodné pro aplikace, které vyžadují časté aktualizace, například online videohry, chatovací aplikace nebo burzovní stránky vývoje cen akcií. Obousměrná komunikace otevřela nová pole možností, díky kterým mohou fungovat efektivně aplikace jako například aplikace pro úpravu dokumentů více uživateli zároveň nebo kvízové aplikace. (Wang, 2013), (Pavlík, 2022)

5.3.7 TinyMCE

TinyMCE je webový editor WYSIWYG určený především k úpravě HTML kódu. Je vydaný pod licencí MIT a dostupný ve všech moderních webových prohlížečích. Zdrojový kód editoru

je otevřený a volně dostupný. Editor je oblíbený díky své jednoduchosti, možnostem vlastních úprav nebo dostupnosti. Při vývoji editoru se bralo v potaz i použití v mobilních zařízeních. Pro uživatele jsou dostupné rozsáhlé možnosti úprav, od běžných změn fontů, barev, zarovnání až po vkládání dynamických objektů nebo úprav přímo HTML a CSS kódu. Podporovány jsou i nejrůznější pluginy, zahrnující pluginy s otevřeným i uzavřeným zdrojovým kódem. Dostupné je také API pro usnadnění implementace vlastních funkcionalit. Tento editor má každý rok přes 350 milionů stažení. (Tiny Technologies Inc., 2024)



Obrázek 7: Rozhraní editoru TinyMCE

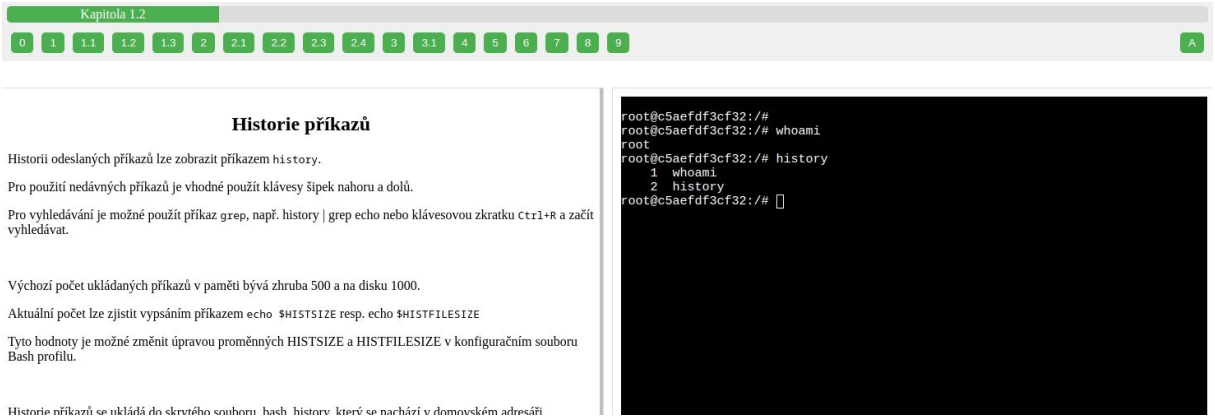
5.4 Webová aplikace

V této části bakalářské práce je představeno rozhraní webové aplikace a ukázka použití již představených softwarových nástrojů.

Pro vytvoření webové aplikace byl použit minimalistický webový framework Node.js Express.

5.4.1 Uživatelská část

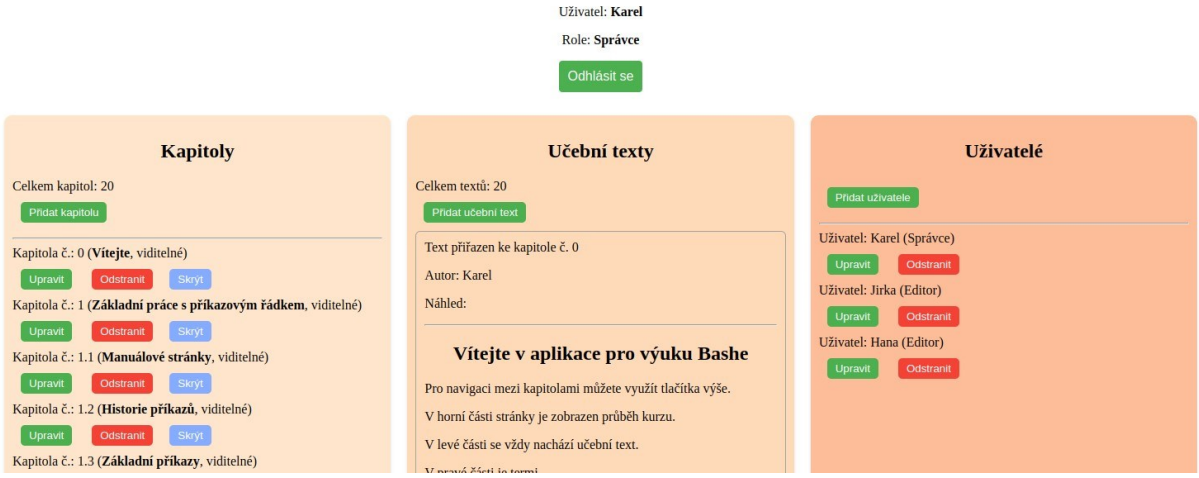
Hlavní okno webové aplikace je rozděleno na dvě poloviny. Nad těmito okny je umožněn v horní části výběr kapitol. Součástí výběru kapitol je i ukazatel průběhu celého kurzu.



Obrázek 8: Uživatelské rozhraní

5.4.2 Administrace

Administrace webové aplikace je dostupná po přihlášení na URL cestě „/management“ nebo pomocí tlačítka vpravo v uživatelské části. Pokud není správce nebo editor přihlášen, je automaticky přesměrován na přihlašovací stránku. Správci či editoři se přihlašují pomocí přihlašovacího jména a hesla.



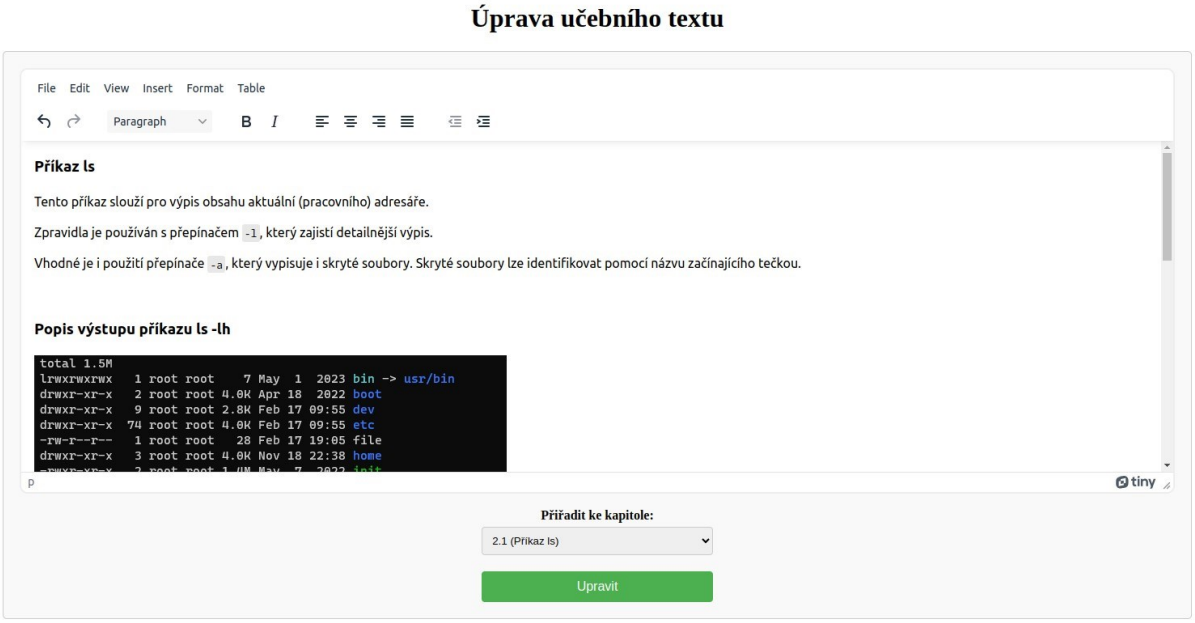
Obrázek 9: Rozhraní administrace (v roli správce)

Rozhraní administrace je rozděleno do tří sloupců. V prvním sloupci se nachází správa jednotlivých kapitol a podkapitol, v druhém učební texty a v posledním uživatelé, správci a editoři. V horní části rozhraní se nachází přihlašovací jméno uživatele, jeho role a možnost odhlásit se z administrace. Jednotlivé kapitoly je možné přidávat, upravovat a odstraňovat. Také je dostupná volba zviditelnění, respektive skrytí kapitoly v uživatelské části. Volby přidávání, úpravy a odstranění jsou také umístěny u učebních textů. Při vytváření učebního textu je každý text přiřazen k vybrané kapitole, pozdější přeřazení k jiné kapitole je možné po stisknutí tlačítka „Upravit“.

Uživatele s přístupem k administraci může spravovat pouze správce. Editoři pouze vidí své kolegy, nemohou u nich nic upravovat. Stejně jako v případě kapitol a učebních textů, správce může přidávat, upravovat a odstraňovat veškeré uživatele. Veškerá hesla jsou bezpečně ukládána do databáze. Před uložením hesla do databáze je vytvořen otisk hesla (hash), který znemožní přečtení hesla v prostém textu v případě napadení a odcizení údajů z databáze. Pro vytvoření otisku hesla je použita knihovna „bcrypt“ a její funkce „hash“. Pomocí funkce „genSalt“ je vygenerovaná sůl, která zajistí unikátní otisk hesla i v případě, že by měli uživatelé stejná hesla. V případě úniku otisků hesel je pro útočníka složitější odvodit heslo v prostém textu dle předem vygenerovaných otisků hesel.

5.4.2.1 Úprava učebního textu

Na stránce pro přidání i úpravu učebního textu byl editor TinyMCE rozšířen o podporu vkládání tabulek a obrázků. Ve spodní části stránky je možné vybrat si ze seznamu dostupných kapitol a učební text přeřadit.



Obrázek 10: Ukázka úpravy učebního textu

Obrázek 11: Ukázka formuláře na přidání nové kapitoly

5.4.3 Databáze

Pro tuto aplikaci byla zvolena databáze SQLite zejména díky své lehkosti a provozu v jediném souboru bez nutnosti správy serveru. V této aplikaci převažuje čtení dat z databáze (načítání učebních textů a kapitol), které je z SQLite databáze velice rychlé díky více vláknovému čtení a provozu databáze jako soubor. Zápis probíhá pouze jedno vláknově, což ale není při převažujícím čtení dat pro tuto aplikaci omezující i do budoucna. (Kolonko, 2018)

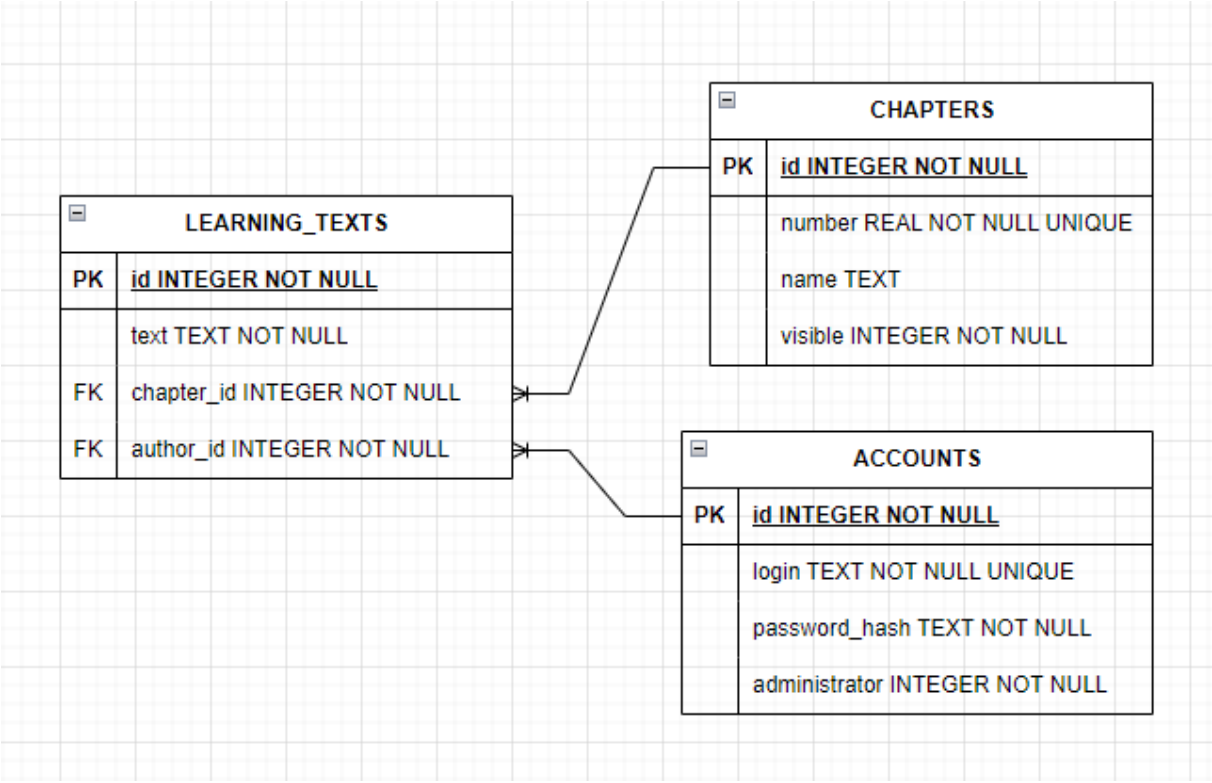
Databáze pro výukovou aplikaci obsahuje celkem tři tabulky.

První tabulka s názvem „ACCOUNTS“ eviduje účty pro přístup ke správě aplikace. Evidováno je unikátní ID pro pozdější propojení s další tabulkou. Dále je evidováno unikátní přihlašovací jméno, pouze otisk hesla z důvodu bezpečnosti a role, zda má uživatel vyšší práva (správce) nebo omezená (editor kapitol a učebních textů).

Druhá tabulka „CHAPTERS“ obsahuje informace o jednotlivých kapitolách. Evidováno je unikátní ID, číslo kapitoly, které může být i desetinné (podkapitola), nepovinně název kapitoly a jako poslední identifikátor, zda má být kapitola zveřejněna nebo skryta koncovým uživatelům.

Poslední tabulkou je tabulka s názvem „LEARNING_TEXTS“. Tato tabulka obsahuje učební texty a dále eviduje ID kapitoly, ke kterému má být učební text přiřazen a ID účtu autora učebního textu.

Následuje obrázek databázového modelu, který byl vytvořen v online aplikaci draw.io:



Obrázek 12: Databázový model (ERD)

5.4.3.1 Připojení databáze

Souborová databáze SQLite s názvem „db.sqlite3“ je v projektu uložena v samostatné složce „data“. Následujícím kódem je importována knihovna SQLite do aplikace:

```
const sqlite3 = require('sqlite3');
```

Slovo „const“ zajišťuje, že do proměnné „sqlite3“ nemůže být později přiřazena jiná hodnota. Na objektu „require('sqlite3')“ je možné zavolat metodu „.verbose()“, která zajistí podrobnější výpis při provádění příkazů. Použití této metody může být užitečné při vývoji a testování.

Následující kód zajišťuje otevření spojení do databáze:

```
const db = new sqlite3.Database('./data/db.sqlite3', sqlite3.OPEN_READONLY, (error) =>
{
   if (error) {
      console.error(error.message);
   }
});
```

V uživatelské části je spojení do databáze otevřeno pouze pro čtení, v administraci aplikace je spojení otevíráno i pro zápis. V případě výskytu jakékoliv chyby je chybová hláška předána do chybového výstupu webové konzole.

5.4.4 Docker obraz

Proces shellu Bash pro uživatele probíhá v samostatném Docker kontejneru pro každou instanci připojení k webu. Při načtení webové stránky je vytvořen nový kontejner a uživateli je umožněn přístup. Díky tomu má každý uživatel vlastní izolovaný prostor. V případě jakékoliv chyby stačí uživateli znovu načíst stránku a bude mu vytvořen nový čistý kontejner. Jako obraz kontejneru je používán oficiální obraz „ubuntu“ v poslední LTS verzi 22.04, navíc doplněn o instalaci textových editorů nano a vim.

Tato část kódu představuje vytvoření a spuštění nového Docker kontejneru:

```
const terminal = pty.spawn('docker', [
      'run',
```

```
    '-it',
    '--rm',
    '-v', `${absolutePath}:/practice:ro`,
    'ubuntu_image',
    'bash'
  ],
    name: 'xterm-color',
  );
```

Pomocí „pty.spawn“ je vytvořen nový pseudoterminál, následně je pomocí metody „write()“ možno posílat příkazy terminálu. Parametr „run“ slouží ke spuštění kontejneru, „-it“ k připojení do terminálu kontejneru, „--rm“ pro odstranění kontejneru po odpojení uživatele, „-v“ pro mapování složky pro procvičovací skripty. Parametr „ubuntu_image“ je název upraveného ubuntu obrazu, „bash“ je příkaz, který je proveden při spuštění kontejneru (spuštění shellu Bash) a „xterm-color“ je volba pro zapnutí barev v terminálu.

5.4.5 Terminál

U kódu vytvoření a spuštění Docker kontejneru je zobrazena deklarace a inicializace objektu „terminal“ na straně serveru. Pomocí metody „write()“ jsou zadané příkazy předávány terminálu, který běží na pozadí. Pro přenos výstupů z terminálu je na objektu nastaven posluchač, který tato data pošle WebSocket spojením. Při ukončení WebSocket spojení (ukončení aplikace) je zavolána metoda „kill()“ pro ukončení terminálu a běhu Docker kontejneru. Na straně klienta je také objekt „terminal“. Je zde provedena inicializace, propojení s HTML elementem, přidána podpora vkládání obsahu, úprava velikosti na základě velikosti okna a několik dalších nastavení.

5.4.6 Procvičovací skripty

Procvičovací skripty jsou uloženy v samostatné složce „data/practice“. Uživateli jsou přístupné přes terminál ve složce „/practice“. Ve složce může uživatel spustit skript pomocí příkazu „./nazev_skriptu“ nebo „bash nazev_skriptu“. Následně je uživatel naváděn pomocí instrukcí vybraného skriptu. Procvičovací skript je možné kdykoliv ukončit pomocí klávesové zkratky „Ctrl+C“.

Do Docker kontejneru je složka se skripty namapována pouze pro čtení (parametr „ro“) s využitím přepínače „-v“ následovně:

```
'-v', `${absolutePath}:/practice:ro`
```

V aplikaci je zadána pouze relativní cesta (./data/practice). Následně pomocí Node.js knihovny „path“ a její metody „resolve()“ je dynamicky vytvořena absolutní cesta a je předána do parametru při spouštění kontejneru.

Procvičovací skript je zpravidla dostupný po každém hlavním tématu. V terminálu se do složky se skripty uživatel dostane pomocí příkazu „cd /practice“. Při nesprávné odpovědi není uživateli rovnou zobrazena správná odpověď, uživatel je naváděn k vlastnímu vyhledání informací, například použití vestavěných manuálových stránek.

```
Zadejte příkaz, který zobrazí obsah adresáře ve formátu dlouhého výpisu:
ls -l
Příkaz 'ls -l' je správně!

Zadejte příkaz, který zobrazí všechny soubory v adresáři, včetně skrytých:
ls

Nápověda: Skryté soubory začínají tečkou, informace lze nalézt v manuálových stránkách příkazu 'ls'
To není úplně správně. Zkuste to znovu:

Stále to není správně. Zde je správná odpověď.
Správný příkaz je: ls -a
```

Obrázek 13: Příklad procvičovacího skriptu

5.5 Možná budoucí vylepšení webové aplikace

Díky přístupu k terminálu se shellem Bash je možné procvičovat i další příkazy, služby či konfiguraci aplikací. Následující možná budoucí vylepšení se týkají především práce s učebními texty.

5.5.1 Zamykání

Do administrace aplikace může mít přístup více správců či editorů zároveň. Při úpravě zejména učebních textů může nastat situace, kdy dva nebo i více uživatelů bude upravovat stejný učební text. Při ukládání úprav může nastat situace, při které uživatel, který uloží text později, přepíše úpravy provedené předešlým uživatelem. Tato situace má více možností řešení. Jedním z řešení je přidat do databáze sloupec (tabulku), která bude udržovat informaci (zámek), že záznam je upravován. Před umožněním úprav je zkontrolováno, zda není záznam uzamčen. Při použití

zamykání je nutné myslet na nestandardní situace, například pokud uživatel opustí aplikaci v průběhu provádění úprav záznamu. To lze řešit odstraněním zámku po určité době neaktivity.

5.5.2 Verzování

Mezi další vylepšení může patřit verzování jednotlivých učebních textů. Při jakékoli úpravě učebního textu by byla vytvořena nová verze a uložena do databáze. Následně by bylo možné procházet a revidovat jednotlivé změny v různých verzích. Verzování je možné použít i jako náhradu zamykání. Učební text bude moci upravovat více uživatelů zároveň, po uložení úprav je pro každou úpravu od uživatele vytvořena nová verze. Následně je možné vybrat (zkombinovat) konkrétní verzi. Dále se nabízí použití verzování při publikaci učebních textů koncovým uživatelům. Nová verze může být uživatelům publikována až po předchozí kontrole a schválení.

ZÁVĚR

Cílem této bakalářské práce bylo vytvořit webovou aplikaci pro výuku skriptovacího jazyka Bash. Webová aplikace nejdříve seznámí uživatele se základními příkazy, dále postupně s příkazy souborového systému, se vstupy a výstupy procesu, příkazy pro zpracování textu a s dalšími jinými příkazy. V další části je uživatel seznámen s tím, jak vytvořit skript, jak ho spustit a následně jsou představeny struktury používané ve skriptech již známé z jiných programovacích jazyků. Uživatel má ve webové aplikaci k dispozici terminál s procesem Bash. Díky tomu se může uživatel používat Bash, i když ho na svém zařízení nemá nainstalovaný. Zároveň také uživatel pouze nečte jen učební text, může rovnou zkoušet praktické příkazy nebo skripty a rovnou má odezvu od Bashe v reálném čase. Pro procvičení naučených znalostí má uživatel k dispozici procvičovací skripty. Tyto skripty se snaží navést uživatele k dohledání informací v případě nesprávně odpovědi, namísto okamžitého zobrazení odpovědi. V případě opakované nesprávné odpovědi je uživatel nakonec zobrazena správná odpověď. Webová aplikace také obsahuje administraci, kde po autentizaci mohou privilegovaní uživatelé přidávat, upravovat, zveřejňovat a případně odstraňovat jednotlivé kapitoly a učební texty. Díky terminálu s Bashem a možnosti přidávat učební texty je možné tuto aplikaci používat do budoucna například na výuku konfigurace služeb či ji použít v jiných částech Unixových operačních systémů.

POUŽITÁ LITERATURA

Bash changes, 2022. ITS - Technology Infrastructure Services [online]. [cit. 2024-02-10]. Dostupné z: https://tiswww.case.edu/php/chet/bash/CHANGES

BERTRAM, Adam, 2017. Discover the power of Bash on Windows [online]. [cit. 2024-02-11]. Dostupné z: https://www.infoworld.com/article/3178631/discover-the-power-of-bash-on-windows.html

BILLEMONT, Maarten, 2023. The Bash Guide [online]. [cit. 2024-03-15]. Dostupné z: https://guide.bash.academy/expansions/

BLAKE, Charles, c1994–2024. Who: Print who is currently logged in [online]. 8.32. GNU coreutils [cit. 2024-03-09]. Dostupné z: https://www.gnu.org/software/coreutils/manual/html_node/who-invocation.html

BOTH, David, 2020. A sysadmin's guide to Bash scripting [online]. [cit. 2024-04-15]. Dostupné z: https://opensource.com/sites/default/files/gated-content/a_sysadmin_s_guide_to_bash_scripting.pdf

DOCKER INC., c2013-2024. How Compose works [online]. [cit. 2024-03-12]. Dostupné z: https://docs.docker.com/compose/compose-application-model/

ECLIPSE FOUNDATION AISBL, 2024. Eclipse ShellWax [online]. [cit. 2024-04-20]. Dostupné z: https://github.com/eclipse/shellwax

Features Of SQLite [online], 2023. [cit. 2024-03-06]. Dostupné z: https://www.sqlite.org/features.html

FOX, Brian a Ramey CHET, c1994–2024. Echo: Print a line of text [online]. 8.32. GNU coreutils [cit. 2024-03-07]. Dostupné z: https://www.gnu.org/software/coreutils/manual/html_node/echo-invocation.html

FREE SOFTWARE FOUNDATION, INC., 2019. Grep [online]. [cit. 2024-03-02]. Dostupné z: https://www.gnu.org/software/grep/manual/grep.html

GRANLUND, Torbjorn, David MACKENZIE a Jim MEYERING, c1994–2024. Cp: Copy files and directories [online]. 8.32. GNU coreutils [cit. 2024-03-09]. Dostupné z: https://www.gnu.org/software/coreutils/manual/html_node/cp-invocation.html

HENRY-STOCKER, Sandra, 2021. Choosing and changing your Linux shell: If you don't love the shell you're using on your Linux system, change it! There are plenty, including bash, fish, ksh, tcsh, zsh. [online]. [cit. 2024-02-11]. Dostupné z: https://www.networkworld.com/article/970162/choosing-and-changing-your-linux-shell.html

HERBST, Michael, 2017. Advanced Bash Scripting. http://doi.org/10.5281/zenodo.1045332.

HOLT, Alan a Chi-Yu HUANG, 2018. Overview of GNU/Linux. Springer, Cham. ISBN 978-3-319-72976-3.

ILIEV, Bobby, 2021. Introduction to Docker [online]. [cit. 2024-04-25]. Dostupné z: https://linuxhandbook.com/content/files/2023/04/introduction-to-docker-light.pdf

JAIN, Hemant, 2022. Recommended Bash Scripting Extensions for VS Code [online]. [cit. 2024-04-20]. Dostupné z: https://scribe.rip/devops-and-sre-learning/recommended-bash-scripting-extensions-for-vs-code-67c62a132978

JETBRAINS S.R.O., c2000-2024. BashSupport Pro [online]. [cit. 2024-04-21]. Dostupné z: https://plugins.jetbrains.com/plugin/13841-bashsupport-pro

KAMENÍK, Pavel. Příkazový řádek v Linuxu: praktická řešení. Brno: Computer Press, 2011. ISBN 978-80-251-2819-0

KARRYS, Luke, Michael RIENSTRA, Myles BORINS a Edward THOMSON, 2023. About npm [online]. [cit. 2024-03-08]. Dostupné z: https://docs.npmjs.com/about-npm

KERRISK, Michael, 2020. Man-pages – conventions for writing Linux man pages. 5.10.

KIM, Byoung Soo, Sang Hyeop LEE, Ye Rim LEE a Jongpil JEONG, 2022. Design and Implementation of Cloud Docker Application Architecture Based on Machine Learning in Container Management for Smart Manufacturing [online]. 12 [cit. 2024-03-12]. Dostupné z: https://doi.org/10.3390/app12136737

KOLONKO, Kamil, 2018. Performance comparison of the most popular relational and non-relational database management systems [online]. [cit. 2024-04-25]. Dostupné z: https://www.diva-portal.org/smash/get/diva2:1199667/FULLTEXT02.pdf

MACKENZIE, David, 2024. Mkdir - make directories. 8.32. GNU coreutils.

MALEKI, Matin, 2022. Shell Scripting Basics [online]. [cit. 2024-03-05]. Dostupné z: https://pressbooks.senecacollege.ca/uli101/chapter/shell-scripting-basics/

MEYERING, Jim a Paul EGGERT, 2023. Comparing and Merging Files [online]. [cit. 2024-03-04]. Dostupné z: https://www.gnu.org/software/diffutils/manual/diffutils.html

MEYERING, Jim, 2020. Sed, a stream editor [online]. [cit. 2024-03-04]. Dostupné z: https://www.gnu.org/software/sed/manual/sed.html

MEYERING, Jim, c1994–2024. Pwd: Print working directory [online]. 8.32. GNU coreutils [cit. 2024-03-05]. Dostupné z: https://www.gnu.org/software/coreutils/manual/html_node/pwd-invocation.html

MINNICK, Chris, 2023. JavaScript All-in-one for Dummies. John Wiley. ISBN 978-1-119-90683-1.

MLYNARIK, Richard, c1994–2024. Whoami: Print effective user name [online]. 8.32. GNU coreutils [cit. 2024-03-06]. Dostupné z: https://www.gnu.org/software/coreutils/manual/html_node/whoami-invocation.html

OPENJS FOUNDATION, 2024. Node.js JavaScript runtime [online]. [cit. 2024-03-08]. Dostupné z: https://nodejs.org/en/about

PARKER, Mike, David MACKENZIE a Jim MEYERING, c1994–2024. Mv: Move (rename) files [online]. 8.32. GNU coreutils [cit. 2024-03-14]. Dostupné z: https://www.gnu.org/software/coreutils/manual/html_node/mv-invocation.html

PAVLÍK, Josef, 2022. Komunikace webové aplikace se serverem pomocí WebSocket [online]. [cit. 2024-04-14]. Dostupné z: https://www.root.cz/clanky/komunikace-webove-aplikace-se-serverem-pomoci-websocket/

RAMEY, Chet, 1998. Bash reference manual. 15. Network Theory Limited.

RANDAL K., Michael, 2011. Mastering Unix Shell Scripting: Bash, Bourne, and Korn Shell Scripting for Programmers, System Administrators, and UNIX Gurus. [online]. John Wiley [cit. 2024-02-14]. Dostupné z: https://ds.amu.edu.et/xmlui/bitstream/handle/123456789/11856/Mastering%20UNIX%20Shell%20Scripting.PDF

ROBBINS, Arnold a David MACKENZIE, c1994–2024. Id: Print user identity [online]. 8.32. GNU coreutils [cit. 2024-03-06]. Dostupné z: https://www.gnu.org/software/coreutils/manual/html_node/id-invocation.html

ROSSI, Alessandro, 2019. Linux streams, what? STDIN, STDOUT, STDERR. /var/blog ~ A blog made to share knowledge [online]. [cit. 2024-02-19]. Dostupné z: https://varslashblog.com/linux-streams-what-stdin-stdout-stderr/

RUBIN, Paul, Arnold ROBBINS, David MACKENZIE a Jim KINGDON, c1994–2024. Touch: Change file timestamps [online]. 8.32. GNU coreutils [cit. 2024-03-11]. Dostupné z: https://www.gnu.org/software/coreutils/manual/html_node/touch-invocation.html

RUBIN, Paul, David MACKENZIE, Richard STALLMAN a Jim MEYERING, c1994–2024. Rm: Remove files or directories [online]. 8.32. GNU coreutils [cit. 2024-03-10]. Dostupné z: https://www.gnu.org/software/coreutils/manual/html_node/rm-invocation.html

SATHEESH, Mithun, Bruno Joseph D'MELLO a Jason KROL, 2015. Web development with MongoDB and NodeJs [online]. Packt Publishing [cit. 2024-03-09]. ISBN 978-1-78528-752-7. Dostupné z: http://www.digitalbreakdown.net/sandbox/Ebooks/Web-Development-with-MongoDB-and-NodeJS.pdf

SMITH, Ben, 2015. Beginning JSON. doi:10.1007/978-1-4842-0202-9.

SOURCELAIR, 2024. A terminal for the web [online]. [cit. 2024-03-08]. Dostupné z: https://github.com/xtermjs/xterm.js

SRIVASTAV, Prakhar, 2023. A Docker Tutorial for Beginners [online]. [cit. 2024-03-12]. Dostupné z: https://docker-curriculum.com/

STALLMAN, Richard a David MACKANZIE, c1994-2024. List directory contents [online]. [cit. 2024-03-12]. Dostupné z: https://www.gnu.org/software/coreutils/manual/html_node/ls-invocation.html

Stdin, stdout, stderr - standard I/O streams [online], 2017. [cit. 2024-03-01]. Dostupné z: https://linux.die.net/man/3/stdin

The GNU Awk User's Guide [online], 2020. [cit. 2024-03-04]. Dostupné z: https://www.gnu.org/software/gawk/manual/gawk.html

TINY TECHNOLOGIES INC., 2024. The world's #1 JavaScript library for rich text editing. Available for React, Vue and Angular [online]. [cit. 2024-03-07]. Dostupné z: https://github.com/tinymce/tinymce

VAUGHT, Andy, 1997. Introduction to Named Pipes [online]. [cit. 2024-03-02]. Dostupné z: https://dl.acm.org/doi/fullHtml/10.5555/326952.326970

WANG, K. C., 2018. Process Management in Unix/Linux. Springer, Cham. ISBN 978-3-319-92429-8. https://doi.org/10.1007/978-3-319-92429-8_3.

WANG, Vanessa, Frank SALIM a Peter MOSKOVITS, 2013. The Definitive Guide to HTML5 WebSocket. 1. 208 s. ISBN 978-1-4302-4741-8. https://doi.org/10.1007/978-1-4302-4741-8.

WATSON, Colin, 2024. Man - an interface to the on-line reference manuals [online]. [cit. 2024-02-29]. Dostupné z: https://manpages.ubuntu.com/manpages/trusty/en/man1/man.1.html

What is Free Software, c1996-2002, 2004-2019, 2021-2024. The GNU Operating System [online]. [cit. 2024-01-31]. Dostupné z: https://www.gnu.org/philosophy/free-sw.html

WHITTAL, Hamish, 2005. Stdin, stdout, stderr [online]. [cit. 2024-03-01]. Dostupné z: https://www.learnlinux.org.za/courses/build/shell-scripting/ch01s04.html

SEZNAM PŘÍLOH

Příloha A – diagram závislostí

Příloha B – přihlašovací údaje k administraci aplikace

Přihlašovací jméno	Heslo	Role
Eva	TXb<<Ql*/P<A4Ym{02}k	správce
Jindřich	eVDRl3"4k0GV~{{Kec`)	editor

Příloha C – zdrojový kód webové aplikace

Tato příloha obsahuje zdrojový kód webové aplikace v archivu s názvem MandikM_WebovaAplikace_SN_2024.zip.