Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Diplomová práce

Výuka základů práce v Bashi pomocí problémové výuky

Vypracoval: Bc. Jakub Novotný

Vedoucí práce: Mgr. Václav Šimandl, Ph.D.

České Budějovice 2018

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 sb. V platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze stag provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 sb. Zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 24. 4. 2018

Bc. Jakub Novotný

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jakub NOVOTNÝ**
Osobní číslo:	**P16338**
Studijní program:	**N7503 Učitelství pro základní školy**
Studijní obory:	**Učitelství technické výchovy pro 2. stupeň základních škol**
	Učitelství informatiky pro 2. stupeň základních škol
Název tématu:	**Výuka základů práce v Bashi pomocí problémové výuky**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Výuka práce v interpretu příkazů Bash je často zaměřena na pouhé reprodukování příkazů, které jsou studentovi ukázány. Takovýto přístup mnohdy studentům nedovoluje pochopit princip používání Bashe a práce v Linuxu obecně. Cílem diplomové práce je vytvořit sadu úloh, které se budou zaměřovat na principy používání linuxového interpretu příkazů Bash a základy programování v něm. Sada úloh by měla být použitelná při výuce studentů informatického zaměření na specializované střední či vysoké škole. Jednotlivé úlohy budou využívat prvky problémové výuky a studenti by v jejich rámci měli řešit určitý problém netriviální povahy. Úlohy se tedy nebudou zaměřovat na použití určitého základního příkazu, ale spíše na výuku konceptů, které jsou typické pro práci v Bashi (např. přesměrování vstupu a výstupu, problematika přístupových práv apod.) a programování v něm (např. specifika používání proměnných).

Vytvořenou sadu úloh student ověří ve výuce na vybrané střední či vysoké škole, toto nasazení vyhodnotí a případně provede úpravu vytipovaných úloh. Student vytvoří manuál pro učitele, kde bude detailně popsáno, k výuce jakého konceptu je každá z úloh zaměřena, jak by ji měl učitel ve výuce použít a jaké jsou nejčastější chyby studentů při jejím řešení.

Rozsah grafických prací: CD ROM

Rozsah pracovní zprávy: 60

Forma zpracování diplomové práce: tištěná

Seznam odborné literatury:

1. BURTCH, Ken O. Linux shell scripting with Bash. Indianapolis: Sams Publishing, 2004. ISBN 0-672-32642-6.
2. DUŠEK, Pavel. Tvorba výukových materiálů pro výuku Linuxu. České Budějovice: Jihočeská univerzita v Českých Budějovicích, 2009. Bakalářská práce.
3. KLIČKOVÁ, Marie. Problémové vyučování ve školní praxi. Praha: Státní pedagogické nakladatelství, 1989. ISBN 80-04-23522-0.
4. MAŇÁK, Josef. Alternativní metody a postupy. Brno: Masarykova univerzita, 1997. ISBN 80-210-1549-7.
5. PETTY, Geoffrey. Moderní vyučování. 6., rozš. a přeprac. vyd. Praha: Portál, 2013. ISBN 978-80-262-0367-4.
6. SHAH, Steve a Wale SOYINKA. Administrace systému Linux: překlad čtvrtého vydání. Praha: Grada, 2007. ISBN 978-80-247-1694-7.
7. SIEVER, Ellen. Linux v kostce: pohotová referenční příručka. Praha: Computer Press, 1999. ISBN 80-7226-227-0.

Vedoucí diplomové práce: **Mgr. Václav Šimandl, Ph.D.**
Katedra informatiky

Datum zadání diplomové práce: **24. dubna 2017**

Termín odevzdání diplomové práce: **30. dubna 2018**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 24. dubna 2017

Abstrakt

Výuka práce v interpretu příkazů Bash je často zaměřena na pouhé reprodukování příkazů, které jsou studentovi ukázány. Takovýto přístup mnohdy studentům nedovoluje pochopit princip používání Bashe a práce v Linuxu obecně a takováto výuka může následně studenty motivovat k pouhému naučení se příkazů nazpaměť bez jakéhokoliv pochopení látky do hloubky. Žáci se tak nenaučí, jak příkaz funguje, jak a kdy ho lze používat v jiných situacích než pouze v ukázkových, kde ho pouze opsali a vše fungovalo. Nemluvě o složitějších skriptech, ve kterých je nutná kombinace několika příkazů s argumenty v přesné posloupnosti, aby skript vykonal požadované operace.

Proto hlavním cílem této diplomové práce je vytvořit sadu úloh, která zaměřuje na principy používání linuxového interpretu příkazů Bash a základy programování v něm. Sada úloh by měla být použitelná při výuce studentů informatického zaměření na specializované střední či vysoké škole. Jednotlivé úlohy využívají prvky problémové výuky a studenti v jejich rámci řeší problémy netriviální povahy. Úlohy se tedy nezaměřují na použití určitého základního příkazu, ale spíše na výuku konceptů, které jsou typické pro práci v Bashi (např. Přesměrování vstupu a výstupu, problematika přístupových práv apod.) A programování v něm (např. Specifika používání proměnných).

Druhým, avšak neméně důležitým, cílem bylo ověření vytvořené sady úloh ve výuce na střední škole pro získání co nejvíce potřebných poznatků k jejich vyhodnocení. Toto nasazení bylo následně vyhodnoceno a byla případně provedena úprava vytipovaných úloh, ve kterých byl shledán větší problém.

Na závěr práce byl vytvořen manuál pro učitele, ve kterém je detailně popsáno, na výuku jakého konceptu je každá z úloh zaměřena, jak by ji měl učitel ve výuce použít a jaké jsou nejčastější chyby studentů při jejím řešení, aby učitel využívající tuto sadu úloh měl s její aplikací ve své výuce co nejméně práce a dokázal se na ni snáze připravit.

Klíčová slova

Linux, Bash, programování, skripty, problémová výuka

Abstract

Problem teaching in the Bash command interpreter is often aimed at simply reproducing the commands that are shown to a student. Such an approach often prevents students from understanding the principle of using Bash and working in Linux in general, and such instruction can then motivate students to simply learn the commands by memorizing without any understanding of the substance in depth. Pupils do not learn how the command works, how and when it can be used in situations other than in the sample, where it was just typed and everything worked. Not to mention more complex scripts that require a combination of several commands with arguments in the exact sequence to make the script perform the required operations.

Therefore, the main goal of this diploma thesis is to create a set of tasks that is focused on the principles of using the Bash command line interpreter and the basics of programming in it. A set of tasks should be useful in teaching informatics at a specialized secondary school or college. Individual tasks use the elements of problem education and students will solve problems of non-trivial disposition within them. Therefore, the tasks are not focused on the use of a basic command, but rather on teaching the concepts typical of Bash (eg. Redirecting outputs) and programming (eg. The specifics of using variables).

The second, but not less important, goal was to verify the set of tasks in the secondary school education in order to obtain the most needed knowledge to evaluate them. This deployment was afterwards evaluated and, if appropriate, the editing of the selected problematic tasks was performed.

At the end of the thesis, a manual for teachers was developed, in which is described in detail, of which of the tasks to teach the concept, how the teacher should use it and what are the most common mistakes of the students in solving it, so that a teacher using this set of tasks has with its application in their education as little work as possible and could easily prepare for it.

Keywords

Linux, Bash, programming, skripts, problem teaching

Poděkování

Mnohokrát děkuji svému vedoucímu diplomové práce panu Mgr. Václavu Šimandlovi, Ph.D. za jeho trpělivost ve vedení mé práce do zdařilého konce, za poskytnutí cenných rad a poznatků při její tvorbě a také za vstřícnost při potřebných konzultacích ve chvílích, kdy zbývalo již málo času. Dále bych rád poděkoval za podporu a silné nervy během tvorby této práce mé přítelkyni Dominice Příhodové a všem ostatním, díky kterým jsem se dostal až k dokončení tohoto díla.

Obsah

1 Úvod

Nejen linuxových serverů, ale celkově serverů, je v dnešní době nesčetné množství, proto je nutné mít i dostatek vzdělaných lidí, kteří budou schopni tyto servery spravovat. Samozřejmě, že si každý může zaplatit různá školení, po kterých administrace nějakého serveru může být o něco jednodušší, ale proč nezačít již na střední škole při výuce. Často se totiž výuka nese pouze v duchu procvičování na vzorových příkladech, ke kterým lze velmi jednoduše naleznout řešení na internetu nebo v učebnicích. Samotná výuka pak ztrácí na významu a žáci se nemusí naučit tolik, protože pouze bezhlavě opisují kód, kterému nerozumí. V této práci se však budeme zabývat pouze Linuxovým operačním systémem a Bashem.

Zaměření této práce je tedy hlavně na to, jak lze předat znalosti žákům středních škol jinak, než tak, aby vše pouze opsali. Toto lze vyřešit změnou metody vyučování. Například zavedením problémové výuky do vyučování, kdy všichni žáci dostanou úlohu, ve které není hned zřejmé, jak ji vyřešit a musí daný problém zkoumat a snažit se najít řešení. Zda k tomu použijí internet či na základě již dříve získaných znalostí budou schopni odvodit řešení, respektive ho vymyslet, je již čistě na nich. Důležité je, že nad problémem budou muset přemýšlet a snažit se najít jeho řešení.

Problémové úlohy mohou do výuky přinést mnoho zajímavých situací. Jakmile se žákovi podaří příklad vyřešit, má z toho mnohem větší radost, než když pouze něco opsal z tabule a začalo mu to fungovat. S tím samozřejmě roste chtíč pracovat na dalším příkladu. Jistě, že tato metoda má i nějaká úskalí. Jmenovat můžeme například to, že někteří žáci rovnou zavrhnou pracovat, protože neví, jak problém vyřešit a nemají chuť se tím dále zabývat. Raději by řešení opsali a následně se to naučili. Jedním z největších úskalí je však příprava učitele na výuku. Aby příklady byly zajímavé, pak učitel na přípravě stráví velmi mnoho času, i na příkladech, které žáci vyřeší během několika málo okamžiků. Avšak díky vyřešenému problému se žáci naučí velmi mnoho z dané problematiky, proto je i pro učitele velmi příjemné sledovat, jak žáci řeší jím vymyšlený problémový příklad a postupně mají radost z postupu, a navíc mají chuť řešit další a další před ně postavené výzvy.

1.1 Cíle práce

Cílem diplomové práce je vytvořit sadu úloh, které se zaměřují na principy používání linuxového interpretu příkazů Bash a základy programování v něm. Sada úloh by měla být použitelná při výuce studentů informatického zaměření na specializované střední či vysoké škole. Jednotlivé úlohy využívají prvky problémové výuky a studenti by v jejich rámci měli řešit určitý problém netriviální povahy. Úlohy se tedy nezaměřují na použití určitého základního příkazu, ale spíše na výuku konceptů, které jsou typické pro práci v Bashi (např. Přesměrování výstupu) a programování v něm (např. Specifika používání proměnných). Vedlejším cílem práce je vytvořenou sadu úloh otestovat ve výuce, zda opravdu učí to, co učit má a aby ji bylo možné použít pro danou problematiku, popřípadě upravit tak, aby byla vyhovující.

Dalším cílem je vytvoření manuálu pro učitele, kde je detailně popsáno, na výuku jakého konceptu je každá z úloh zaměřena, jak by ji měl učitel ve výuce použít, jaké jsou nejčastější chyby studentů při jejím řešení a jaké je bylo od úlohy očekávání a jaká byla následná realita ve výuce.

Tato práce tedy přináší sadu úloh, kterou může využít jakýkoliv učitel ve své výuce zaměřené na základní práce v linuxovém terminálu a samotné programování v interpretu příkazů Bash.

Součástí práce je sbírka úloh, kde se každá úloha zaměřuje na určitou problematiku v Bashi, tedy úlohy jsou poměrně rozmanitého charakteru. Příklady začínají úplným základem, kdy si žáci vyzkouší první otevření a napsání skriptu v Bashi, až po psaní kódu, vyhledávání textu a kombinovaných podmínek.

Vhodné samozřejmě bylo nejen tuto sbírku vytvořit, ale i otestovat ve výuce, kde by se mohly v příkladech vyskytnout chyby nebo kde jsou slabiny. Proto veškeré úlohy ze sady úloh byly otestovány ve výuce ve dvou skupinách a následně změněny k lepšímu, pokud si tak úlohy například zadáním vyžadovaly. Po této části byl zkompletován manuál pro učitele, jaké bylo očekávání, jaká byla výsledná realita a jaké byly problémy s řešením úkolů žáky.

1.2 Metody práce

Prvním krokem této práce bylo studium literatury zaměřené na problémovou výuku, kdy bylo nutné zjistit, jaké možnosti tato metoda výuky obsahuje a jakou úlohu lze považovat za problémovou a jakou již ne. Byly studovány publikace zaměřené na problémovou výuku obecně či publikace, které jsou jakýmsi pomocníkem ve světe Linuxu, eventuálně knihy o programování v Bashi od a do z, kde je zadání příkladu včetně jeho řešení. Průběžně tedy s literaturou zaměřenou na problémovou výuku bylo nutné nastudovat i publikace zaměřené na Linux. I zde je publikací mnoho, nicméně jen velmi těžko lze naleznout nějakou sbírku problémových úloh zaměřených na Linux, natož pak na samotnou výuku v Bashi. Tedy, knihu zaměřenou přímo na problémovou výuku v Bashi se mi však bohužel najít nepodařilo žádnou.

Byla studována literatura o problémové výuce a o Bashi a nejčastěji vyskytnuté příklady z knih byly použity pro výuku, aby se výuka co nejvíce přiblížila praxi v Bashi. Po nalezení těchto důležitých věcí začala samotná tvorba sbírky úloh, aby bylo následně možné vytvořené úlohy vzít do výuky. Po vytvoření sbírky úloh následovala příprava učebny pro výuku a ověření ve výuce. Ověřování ve výuce je poměrně rozsáhlá část, proto má vyčleněnou svou vlastní kapitolu číslo 6.

1.2.1 Vlastní tvorba úloh

Byly připraveny úlohy, které následně v hodinách žáci řešili. Prvotně bylo připraveno několik úloh na ukázku, aby se s řešením úloh žáci seznámili a pochopili, co se od nich bude v následujících hodinách zaměřených již čistě na Bash pomocí problémové výuky vyžadovat. Tyto úkoly byly zaměřené na různé funkcionality v Linuxu. Ze začátku to byly základní funckionality, které si žáci zkoušeli prvotně v terminálu, aby viděli ihned výsledek a neodradilo je to od práce ve skriptování, kde výsledek viditelný ihned není. Tyto příklady jsou uvedeny v sadě úloh pod číslem 1, tedy přípravné úkoly. Je to například změna adresáře, výpis obsahu adresáře, tvorba, kopírování a mazání složek a souborů. Úlohy zaměřené již na Bash jsou rozmanitého charakteru, aby se ověřily získané znalosti a dovednosti žáků.

1.2.2 Příprava žáků na problémovou výuku

Následným krokem byla příprava žáků střední školy na problémovou výuku v Bashi. Jednalo se o třídu studující obor informační technologie, kde třída byla rozdělena na

dvě skupiny po patnácti žácích. Střední školu jsem vybral na základě toho, kde vyučuji operační systémy. Touto školou byla střední odborná škola strojní a elektrotechnická ve Velešíně.

Předchozí znalosti těchto žáků byly zejména základní orientace a práce v systémech Windows a Linux, nicméně žádné složité úkony, skripty a podobně. Hlavně se jednalo o práci v grafickém prostředí, nikoliv v textovém. Výuka v obou skupinách byla obdobná a díky tomu bylo možné otestovat sadu úloh na větším vzorku žáků a následně mezi nimi porovnat chyby či naopak dobrá možná řešení.

U této třídy probíhala příprava žáků v předešlé výuce, kdy žáci pracovali v programech Microsoft Office Word a Excel, následně pracovali s dávkovými soubory. Veškeré úlohy byly zadávány problémově, tedy pro žáky nebyl přechod k Bashi nikterak složitý.

1.2.3 Příprava počítačů k výuce

V této části bych rád pouze poznamenal výčet určitých možností, jak lze s žáky využít sbírku úloh z této práce. I když samozřejmě existuje metod více, já v této práci otestoval tři možnosti, jak s žáky pracovat a rád bych i tuto informaci předal, co se mi osvědčilo ve výuce.

Jedna z možností, jak s žáky pracovat v Linuxu a Bashi, je nainstalování vlastního operačního systému Linux na disk počítače vedle primárního operačního systému a mít možnost mezi nimi přepínat. Při ověřování materiálů se jednalo konkrétně o systém Ubuntu v desktop verzi s grafickým rozhraním, tedy bylo pro žáky jednodušší například vytvořit složku kliknutím myší bez použití příkazu. Samozřejmě se zde nabízí i možnost instalace Ubuntu bez GUI.

Další z možností, která je řekněme obdobou té první, bylo nainstalování Ubuntu do virtuálního počítače. Vesměs se jednalo o podobnu výuku, tedy opět žáci, kteří chtěli některé části úloh obejít, tak se jim to podařilo. Nicméně tato možnost byla poměrně zdlouhavá, protože se s žáky musí připravit (nainstalovat) virtuální počítač a následně systém Linux. Samozřejmě zde je i možnost, že učitel nainstaluje pouze jednu stanici a následně rozkopíruje do ostatních obraz. Nicméně tato varianta byla zvolena pro žáky, aby si také vyzkoušeli instalaci systému Linux. Také je možnost, že tuto práci udělá sám učitel či například připraví správce sítě a učitel již bude

mít jen možnost přijít ke stanicím s žáky a rozdat úkoly. Nicméně instalace operačního systému s žáky může být pouze jako zajímavost ve výuce, aby si to vyzkoušeli i žáci. Nemělo by se jednat o žádnou složitou administraci, nýbrž o čistou instalaci systému pro rozšíření znalostí.

Nejlepším řešením však bylo asi nakonec to poslední vyzkoušené. Nainstalovat Linux na jakýkoliv počítač, vytvořit v něm každému žákovi uživatelský profil s domovským adresářem, ve kterém mohl pracovat a bylo po problémech, kdy si žáci mezi sebou mohli jednoduše posílat data či podvádět použitím například myši. Pomocí vzdáleného terminálu byla jediná možnost, jak začít pracovat a neměli jinou možnost, než používat příkazy. Tedy například vytvořit složku již museli příkazem, protože myš v textovém rozhraní pro ně byla nepoužitelná.

U tohoto řešení můžeme zvolit i jednodušší variantu, kdy budou všichni žáci používat pouze jeden účet a v domovských adresářích si vytvoří každý svou složku se svým jménem a bude pracovat pouze v ní. Nicméně toto řešení lze využít pouze pro výuku, nikoliv pro ověřování znalostí.

Příkaz, kterým lze zajistit přidání uživatele s*udo useradd –m uzivatel*. Dále je nutné nastavit uživatelskému účtu heslo pomocí s*udo passwd uzivatel*. Takto můžeme ručně přidat uživatelů, kolik budeme chtít a díky volbě *–m* se zajistí, že každý uživatelský účet bude mít automaticky vytvořený domovský adresář. Abychom mohli zajistit vzdálený přístup, například přes Putty, musíme nainstalovat OpenSSH. Toto nám zajistí s*udo apt-get install openssh-server –y*. Po zadání tohoto příkazu se nainstaluje OpenSSH. Po instalaci můžeme ještě změnit například port, pod kterým se budeme připojovat k tomuto serveru v konfiguračním souboru */etc/ssh/sshd_config*.

Důležité je také zmínit, jak učitel bude žákům předávat zadání k úlohám, eventuálně skripty, které budou dostupné pro žáky, ve kterých budou například muset objevit a opravit chybu. Možností je samozřejmě opět více, avšak já volil variantu sdíleného disku na školním server, kde byly úlohy očíslovány a rozděleny ve složkách. Každá složka obsahovala buďto zadání se skriptem či pouze zadání. Problém může nastat ve chvíli, kdy žáci budou pracovat ve virtuálních strojích. Pak je přístup ke sdíleným diskům trochu složitější. Buďto se musí virtuální počítač nastavit tak, aby viděl do interní sítě, a tím pádem i sdílené disky nebo zvolit variantu, kdy žáci stáhnou z FTP pomocí příkazu *wget* data do svého pracovního prostředí, například přímo z odkazů

v této práci, kde jsou soubory zazipované, takže si budou žáci muset ještě vyzkoušet rozbalit soubory pomocí příkazu *unzip*.

Kompletní zadání pro žáky včetně vzorového rozdělení s hotovými skripty pro učitele a skripty pro žáky jsou na konci této práce v přílohách.

1.2.3.1 Reflexe výuky

Dle výukového plánu na školní rok byla rozvržená výuka pro dané skupiny a následně v nich probíhala výuka / testování vytvořených příkladů. Po odučení daného tématu bylo zhodnoceno, v čem žáci dělali chyby, co bylo nesrozumitelné, či co pro žáky bylo naopak velmi jednoduché. Po zpracování těchto poznatků byly úlohy popřípadě upraveny tak, aby vyhovovaly a učily danou problematiku lépe, než byly úlohy prvotně vytvořené, eventuálně i následně upravené pro druhou skupinu. Toto vše se odrazilo v závěrečném vyhodnocení sady úloh v závěru práce, což může následně být nápomocné při využití této sady úloh pro jakéhokoliv učitele, který bude tuto sadu chtít použít.

2 Problémové úlohy

Před samotným představením problémových úloh z praktické části této diplomové práce je potřeba definovat, co to problémová úloha, respektive problémové vyučování je. Na toto téma je mnoho definic a nejvíce se liší v samotném použití dle vyučovaného předmětu. Samozřejmostí je i odlišení od ostatních typů úloh, jakým způsobem jsou postaveny, co se očekává o žáků a zejména od vyučujícího.

Jak jsem již zmiňoval, mnohdy se může terminologie v této oblasti určitým způsobem lišit. Stejné typy úloh se mohou v různých publikacích objevovat pod různými názvy. Jmenovat můžeme například úlohy, zajímavé úlohy, cvičení či příklady [1]. V této práci se tedy setkáme s více pojmy, například úloha, problémová úloha, problémová otázka, úkol či problémová situace.

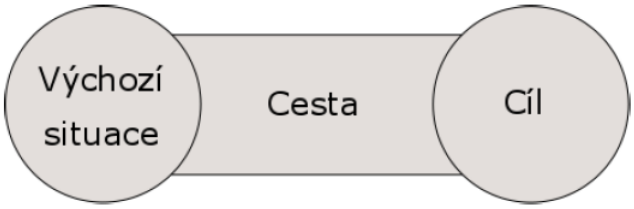
I když se tato práce zaměřuje na výuku zejména pro střední školy, můžeme citovat některé části z Rámcového vzdělávacího programu pro základní školy, zejména klíčové kompetence, rozdělení a definice problémových úloh, a stejně tak i například kompetence k řešení problémů. Můžeme předpokládat, že žák by ze základní školy měl mít takové vzdělání a schopnosti, aby postupným dalším vzděláváním a rozvíjením svých znalostí byl schopen vyřešit úlohy na střední škole. [1] Dokument RVP lze použít jako oporu v definování problémových úloh a řešení.

Rámcový vzdělávací program pro základní vzdělávání mluví o nestandardních problémových úlohách tak, že jejich správné vyřešení není vždy závislé na znalosti žáka dané problematiky, protože by si zde žák měl být schopen poradit i bez této znalosti. A samozřejmě je nutné podotknout, že by tedy žák pro vyřešení těchto typů úloh měl do jisté míry bych schopen uplatnit své logické myšlení. [1]

Problémové úlohy bychom mohli rozdělit do několika skupin, kdy záleží na tom, jakým způsobem je před žáky problém postaven. Zda je to pro žáky zcela ojedinělá věc, se kterou se doposud nesetkali, zda je to problém dosud neviděný, ale lze pomocí logického myšlení uplatnit z dřívějších znalostí určitý typ algoritmu, či je pro žáky tento problém k vyřešení triviální, protože se s něčím podobným již setkali. [2]

Při pohledu na řešení úlohy se musí vzít v potaz to, jaké složky tvoří problém, uvádí Jan kopka [2]. Tyto složky jsou dle něj tři:

1. Výchozí situace = zadání problému, obsahuje veškeré známé informace
2. Cíl = řešení, které chce řešil dosáhnout
3. Cesta = postup, který řešitel využívá, aby se dostal z výchozí situace do cíle



Obrázek 1 znázornění problému [2]

Ze schématu lze odvodit, že řešitel úlohy musí nejdříve pochopit výchozí situaci a na základě tohoto pochopení je poté schopen začít hledat cestu k dosažení cíle. Zadaná situace má jasné zadání a řešitel by tuto úlohu měl pochopit jednoznačně, aby vždy došel ke stejnému cíli, jaký se požaduje. Rozdílnost už ale může být v cestě. Dle toho, jak je pochopeno zadání, respektive výchozí situace, se pak začne řešit postup k cíli. Každý z řešitelů může k cíli dojít jiným způsobem, avšak v cíli by mělo být řešení stejné [2].

Při zadávání problémových situací je možné narazit na několik překážek, nicméně jedna z nejčastějších může být právě ta, kterou popisuje kopka: „*Někdy je těžké určit, zda pro určitého žáka je zadaný problém rutinní či nikoliv. Dostatečným procvičováním postupně přechází určitý druh problému z kategorie nerutinních do kategorie rutinních problémů. Navíc, v danou chvíli může být určitý problém pro některé žáky třídy rutinní a pro jiné nerutinní.*“ [2]

S výše popsanou situací se může setkat totiž každý učitel. Některé úlohy se mohou právě učiteli zdát velmi jednoduché, až triviální, a bude očekávat, že všichni žáci úkol budou mít vyřešený během krátké chvíle. Ale některým žákům mohou i tyto triviální úlohy zabrat dlouhé chvíle, než je pochopí a vyřeší. Zatímco žáci, kteří se již s nějakým podobným problémem setkali, budou tuto úlohu řešit pouze krátkou chvíli. A v návaznosti na tuto úlohu může být další úkol, který by již měl být jednoduchým pro všechny žáky, protože by měli být schopni použít „algoritmus“ naučený z předchozí úlohy, a tak tento další úkol zpracovat v obdobném čase všichni žáci. Když žáci takto úkol vyřeší, v danou chvíli se stane z nerutinního příkladu příklad rutinní.

3 Pojem řešení problémů v kurikulárních dokumentech

Kompletní vzdělávání nejen na základních školách se opírá o Rámcový vzdělávací program pro základní vzdělávání. Rád bych proto popsal, jakou pozici v něm mají tyto nestandardní problémové úlohy a problémové situace. Obsah RVP pro střední školy se zaměřením na informační technologie se ve výsledku v řešených pojmech od základního vzdělávání nijak neliší.

3.1 Rámcový vzdělávací program

RVP ZV definuje, že cílem základního vzdělávání je pomoci žákům utvářet a rozvíjet klíčové kompetence a tím jim poskytnout to nejdůležitější ze všeobecného vzdělávání, které se zaměřuje především na praktické jednání a situace, které jsou blízké reálnému životu: *„klíčové kompetence představují souhrn vědomostí, dovedností, schopností, postojů a hodnot důležitých pro osobní rozvoj a uplatnění každého člena společnosti.“* [1]

V průběhu celého života nás provází osvojování si klíčových kompetencí, počínaje už prvním stupněm základní školy. Samozřejmě je to velmi dlouhý a složitý proces, kterým si však musí projít každý. Tento druh kompetencí se navzájem prolíná různými způsoby, nicméně nejsou součástí konkrétního vyučovaného předmětu, ale lze je získat pouze jako výsledek celkového procesu vzdělávání [2]. A právě k těmto klíčovým kompetencím směřuje veškerý obsah vzdělávání, který ve školách probíhá. „*V etapě základního vzdělávání jsou za klíčové kompetence považovány: kompetence k učení; kompetence k řešení problémů; kompetence komunikativní; kompetence sociální a personální; kompetence občanské; kompetence pracovní.*“ [1]

3.1.1 Kompetence k řešení problémů

Právě jednou ze šesti klíčových kompetencí je kompetence k řešení problémů. O tuto část dokumentu se opírá tato diplomová práce, zejména pak sada problémových úloh. Některé důležité body bych rád ocitoval.

Z dokumentu můžeme vybrat například, že žák:

- *„vnímá nejrůznější problémové situace ve škole i mimo ni, rozpozná a pochopí problém, přemýšlí o nesrovnalostech a jejich příčinách, promyslí a naplánuje způsob řešení problémů a využívá k tomu vlastního úsudku a zkušeností*
- *Vyhledá informace vhodné k řešení problému, nachází jejich shodné, podobné a odlišné znaky, využívá získané vědomosti a dovednosti k objevování různých variant řešení, nenechá se odradit případným nezdarem a vytrvale hledá konečné řešení problému*
- *Samostatně řeší problémy; volí vhodné způsoby řešení; užívá při řešení problémů logické, matematické a empirické postupy*
- *Ověřuje prakticky správnost řešení problémů a osvědčené postupy aplikuje při řešení obdobných nebo nových problémových situací, sleduje vlastní pokrok při zdolávání problémů“* [1]

Na základě tohoto výčtu je jasné, že žáci by měli poznat problém, následně ho pochopit a začít o něm přemýšlet. Jakmile ho začne řešit, měl by být schopen si dohledat informace, které mu brání k vyřešení a zkoušet různé varianty, jak dosáhnout cíle. Jakmile se mu podaří problém vyřešit, měl by si ověřit správnost svého výsledku, zda je opravdu správně.

4 Řešení problémů jako výuková metoda

V dnešním vyučování mají učitele mnoho možností, jakou výukovou metodu zvolit. My se budeme zabývat metodou řešení problémů, jak již název této práce napovídá. Proto bych tedy ještě rád popsal pojem výuková metoda a zařazení metody řešení problémů mezi ostatními metodami.

Pedagogický slovník definuje výukovou metodu jako postup, cestu či způsob vyučování. Učitel si stanoví cíle vzdělávání a pomocí výukové metody vede žáka k dosažení předem vytyčených cílů vzdělávání [3]. Průcha, Walterová a Mareš definují výukovou metodu tak, že ji lze chápat jako způsob, jakým uspořádá učitel činnost svou, ale i žáků, který vede ke splnění stanovených cílů [3].

4.1 Klasifikace výukových metod

Ve vyučování lze použít velké množství výukových metod, což nás vede alespoň k částečnému rozdělení a zařazení právě problémové výuky. Velmi často se v literatuře setkáváme s rozdělením podle Maňáka a Švece, který řeší kritérium postupně se stupňující složitosti, tedy začít něčím jednodušším a postupně zvyšovat náročnost řešení úkolu [4]. Maňák a Švec rozděluje metody na tři skupiny, tedy na klasické, aktivizující a komplexní metody [4]. Další autoři si pro klasifikaci metod vybírají i jiná kritéria, například podle fází vyučovacího procesu, podle způsobu prezentace, podle charakteru specifické činnosti či obecně podle způsobu, jak reagují žáci na otázky učitele [5].

Aktivizující metody		
Slovní	**Názorně-demonstrační**	**Dovednostně-praktické**
Metody diskusní	Metody situační	Metody inscenační
Metody heuristické, řešení problémů		Didaktické hry

Tabulka 1 zařazení metody řešení problémů dle Maňáka [4]

4.2 Problém a problémová situace

Rád bych tyto pojmy rozvedl trochu detailněji a přidal i nějaké příklady. Žáci se často i v běžném životě potýkají s problémy, které je nutné vyřešit. Samozřejmě každý již nějakou zkušenost s řešením problémů má. Denně se setkáváme s problémy či problémovými situacemi, tedy není to pro nás z obecného hlediska až tak velká neznámá.

Jak píše Kličková ve své knize Problémové vyučování ve školní praxi: *„… když je člověk nucen každý den řešit problémy v jeho životě, proč by nebylo možné využít problémových situací i ve školství …“*. [6] Problémová situace má totiž dvě hlavní složky, a to složku obsahovou (předmět problému) a složku osobnostní (motivace žáka) [7]. Můžeme se domnívat, že právě díky této motivaci by u žáka měla vzniknout touha po poznání a vyřešení neznámého problému. To však záleží pouze na tom, jaký typ úlohy učitel zvolí. Pokud bude úloha fádního charakteru, pak žák ztratí zájem tuto úlohu řešit, protože jak předmětová, tak motivační stránka problému neponese vysokou informační hodnotu. Zatímco při kvalitně vytvořené úloze žák ucítí, že předmětová stránka po vyřešení problému mu přinese mnoho informací, které ho v běžném životě mohou posunout dál, pak žák zvýší svou motivaci úlohu vyřešit.

Mnoho učitelů ve školách používá metodu frontální výuky kombinovanou s rozhovorem. Zadání příkladů jsou striktní, a tak žákům nezbývá mnoho prostoru pro jejich seberealizaci a zlepšování logického myšlení, které je do běžného života velmi důležité. I když učitel během rozhovorů je v silné interakci se žákem, žákova vlastní fantazie bývá často zcela potlačena, protože musí odpovědět tak, jak si učitel přeje, aby žák odpověděl. [5] v tuto chvíli již žákům nezbývá mnoho místa pro zlepšení svých nabytých znalostí. Nemá totiž čas na to, aby si mohl klást vlastní otázky a pokusit se najít cestu, kterou by vymyslel své vlastní řešení. I když žák nezná ihned přesnou odpověď, tak se nejedná tak úplně o problém v námi požadovaném slova smyslu. Je to pouze problém pro žáka, který nezná odpověď, ale nemá ani možnost, jak by odpověď vymyslel, protože nemá požadovanou znalost ještě naučenou nazpaměť a praktickým uvažováním ji stejně nevymyslí [6]. Pro názornost si můžeme uvést příklad ze zkoušení z dějepisu nebo například biologie či zeměpisu. Ve chvíli, kdy se učitel zeptá, zda je možné, aby se setkal Karel IV. s Marií Terezií a žák nebude vědět, v jakých letech oba dva žili, pak se jedná o problém, který by měl být žák schopen vyřešit logickým uvažováním.

Problémovou otázkou se otázka stane ve chvíli, když k získání odpovědi nemáme veškeré potřebné informace a musíme nejprve zjišťovat, jaké jsou možnosti, jaké informace chybí a žák musí v danou chvíli hledat cestu k dosažení cíle. Pokud postupně doplňuje informace k výchozí situaci, pak si hledá cestu, kterou cíle dosáhne. Dále také můžeme problém dle Kličkové definovat tak, že žák musí řešit svým aktivním zkoumáním a usilováním překonat obtíže, díky kterým získá nové zkušenosti a poznatky [6].

Pokud se tedy žák dostane do problémové situace, kdy má pocit, že úloha, kterou má vyřešit, je nad jeho znalosti či schopnosti a nadchne ho, pak se do ní s radostí pustí, protože v ní vidí smysl a chce získat nové vědomosti a zkušenosti. To souvisí s tím, že žák v danou chvíli cítí, že sice nezná řešení úlohy, ale při vynaložení určitého úsilí tento problém překoná a vyřeší ho.

4.3 Postup při řešení problémů

Skalková píše, že vyučování a učení může probíhat efektivně pouze ve chvíli, kdy se styl a metody práce ve škole přiblíží co nejvíce situacím z běžného života. Můžeme tedy tvrdit, že díky tomu jsou žáci motivováni k samostatné práci a řešení problémových úloh, protože by se mohli v běžném životě setkat s obdobnou situací, ke které by již díky vyřešení úlohy měli vodítko, jak se s ní vypořádat [5].

Jelikož chceme řešit problém, pak musíme připustit, že problém neobsahuje všechna data potřebná pro vyřešení požadované odpovědi. Jakmile se nastřádají veškeré informace a dojde se cíle, mělo by následovat ověření hypotézy [6]. V následujícím textu popíši, jaké jsou fáze řešení problému.

4.3.1 Objevení problému a jeho analýza

Učitel formuluje pro žáky úlohu tak, aby byla pro žáky zadána problémově. Buďto tak, že problém již učitel přímo zadá v úloze, či si problém musí žák naleznout sám. [5] Příkladem může být již v naší problematice například to, zda žák v příkladu bude muset zjistit, v jaké z podmínek je chyba nebo se před ně postaví zadání takové, že bude žák pouze vědět, že program dělá něco špatně a on sám bude muset pochopit, co by program měl správně dělat, a až po pochopení chybu opravit.

4.3.2 Řešení problému

Za složitou myšlenkovou činnost lze považovat situaci, kdy žák směřuje k řešení problému pomocí vytvoření hypotéz a jejich následném ověřování. Na základě tohoto tvrzení můžeme rozdělit na čtyři formy, kterými lze problém řešit. [6]

4.3.2.1 Pokus omyl

Tuto metodu se žáci pokoušejí využít ve chvíli, kdy mají pocit, že problém je jednoduchý a k jeho vyřešení postačí málo. Buďto se to povede, nebo ne. Nicméně nalezení správného řešení je velmi složité, a to z toho důvodu, že hledání řešení je zcela náhodné. Zejména se pak žák snaží výsledek uhodnout bez jakéhokoliv pochopení úlohy. [6]

4.3.2.2 Vhled nebo postřeh

Žák se postaví před problém a snaží se ho pochopit. V určitou chvíli nastane situace, kdy žák problému náhle porozumí, tím ukončí své objevování a nalezne správnou cestu k řešení, respektive k cíli. Často se můžeme setkat, že tato metoda je doprovázená metodou užití minulé zkušenosti, ačkoliv ne vždy to může být výhodou. [6]

4.3.2.3 Užití minulé zkušenosti

Díky dříve osvojeným návykům a získaným informacím je žák schopen začít řešit problém, který mohl být přinejmenším částečně podobný až stejný. Pokud se již žák s něčím takovým setkal, může tato metoda pro něj být výhodná, ale i destruktivní, protože zde funguje funkční fixace. Žák v minulosti mohl podobný problém vyřešit, ale špatně a toto špatné řešení si zapamatoval a opět ho nyní aplikuje. [6]

4.3.2.4 Rozumová analýza

Část této metody používají všechny předchozí metody, protože se jedná o jakési vnímání situace na základě rozumu daného jedince. Každý má však toto vnímání jiné, proto je se každý žák u stejné příkladu může rozhodnout úplně jinak, protože jeho racionální uvažování je na jiné úrovni. Navíc může být často tato metoda oproti ostatním, například intuitivní metodě, pomalejší. [13]

4.3.3 Kontrola správnosti řešení

Již od prvního stupně se žáci v matematice učí, že po vyřešení slovní úlohy je nutné provést zkoušku, aby si ověřili, že jejich postup řešení je správný. Stejně tak tomu je,

pokud to převedeme do obecného řešení problémových situací a problémů. Každý problém má jiná zadání a jiná řešení. Jakmile dosáhneme cíle, pak bychom měli zkontrolovat, zda odpovídá počátečnímu zadání problému.

Protože tato kontrola plní funkci zpětné vazby, je velmi důležité ji neopomínat. Informuje totiž o špatných krocích v řešení problému a lze díky ní zjistit, kde se stala chyba a proč problém nebyl správně vyřešen. [5] Není však lehké odhalit, jaké metody žák užil, aby úkol vyřešil. Učitel tak může pouze usuzovat na základě žákem vyřešeného úkolu, jaký použil postup a do jaké situace se během řešení dostal.

Ve sbírce úloh máme dva typy kontroly. První je zpětná vazba pro žáky, druhá kontrola je učitelem. Žák svůj postup zkontroluje u některých úloh velmi jednoduše. Skript začne dělat přesně to, co je v zadání napsáno. Dle toho žák zjistí, že skript naprogramoval správně. Příkladem může být například úloha s podivuhodnou kalkulačkou, kdy přímo v zadání má zapsáno, jaké výsledky mají vyjít. Co se týká skriptů, u kterých žák sám bude těžko kontrolovat, zda mu skript vybral opravdu veškerá požadovaná data. Tohoto se může pouze domnívat. V tuto chvíli by měl požádat učitele o kontrolu správnosti. Ten má vzorové řešení v metodických listech, tedy neměl by mít sebemenší problém s kontrolou.

5 Linux

Operační systém Linux není pouhý monolitický program. Jedná se o soubor mnoha programů, díky kterým tvoří z počítače užitečný nástroj. Jeho srdcem je takzvané jádro, které obstarává pouze nejzákladnější operace. K práci však pouhé jádro není dostačující. Proto je nezbytně nutné, aby do systému byly implementovány další programy, které utvoří celek jako fungující, stabilní a kvalitní operační systém. [7]

Linux je založen na operačním systému Unix a postupným přepisováním zdrojového kódu, jedná se o otevřený software, vznikl právě tento systém. Hned o počátku se Linux těšil velkému zájmu jak ze strany programátorů, tak ze strany uživatelů. Každému však pro práci vyhovují jiné programy, a proto je možné vybírat dle potřeb z různých distribucí systému. [8]

Právě tyto distribuce jsou odlišením části nad jádrem, tedy každá distribuce může obsahovat jiné programy, které vyhovují dle potřeb, na který aspekt praxe se zaměřují. Některá může nabídnout jednoduchost instalace a použití, jiná zase co největší množství aplikací v základu. [9]

5.1 Vybrané součásti distribucí Linux

V této části si uvedeme několik vybraných součástí systému Linux, bez kterých bychom mohli pracovat jen velmi těžko, nebo dokonce vůbec.

5.1.1 Linuxové jádro (kernel)

Jedná se o vlastní jádro, které jako jediné umí přistupovat k hardwaru počítače a vykonává úlohy, které uživatel požaduje. Například to může být zobrazení textu či poslání obrázku na tiskárnu. Bez jádra by nemohl však tento systém existovat. [8]

5.1.2 Příkazový interpret (shell)

Součást, která je pro nás v této práci asi nejvíce zajímavá, protože celá praktická část používá právě shell. Díky němu totiž také můžeme ovládat pomocí příkazů operační systém.

Díky shellu je možné komunikovat mezi jádrem a uživatelem. Z praxe je nejznámější interpret Bash (Bourne again shell), který je odvozen od svého předchůdce [8].

Rozhraní shellu je textové prostředí, které se používá k zadávání instrukcí z klávesnice, namísto použití například myši. Interpret načítá zadané příkazy, zpracuje je a následně předá jádru do nejzákladnější úrovně. [9]

5.1.3 Příkazy, nástroje, aplikace

Jedná se o programy, které jsou v systému implementovány a lze je využívat pro různé úkony. Příkladem může být například zobrazení obsahu souboru na obrazovce z těch jednodušších a ze složitějších například zpracování textu, které zahrnuje spousty funkcí. Po napsání příkazu do příkazového interpretu začne probíhat na pozadí nějaký program, který požadované funkce vykonává. [9]

5.1.4 Překladače

Jak název napovídá, jedná se o překládání jazyka, kterému rozumí člověk do počítačového jazyku. Lidé tedy napíšou program v jakémkoliv programovacím jazyce, například v C nebo Java. Daný kód je napsaný tak, aby mu rozuměl člověk a překladač ho transformuje do strojového kódu, kterému zase rozumí počítač a dokáže úkony provést. [9]

5.1.5 Pracovní prostředí

Zjednodušeně se jedná o grafické pracovní prostředí (GUI – Graphical user interface), které je pro většinu běžných uživatelů mnohem přívětivější oproti ovládání systému textovými příkazy. V Linuxu existuje několik těchto prostředí a uživatel si může vybrat, v jakém chce pracovat, které nejvíce vyhovuje jeho potřebám. Dokonce některá z nich se podobají i prostředí systému Windows. Jedná se tedy o prostředí, ve kterém se úkony provádí vybráním položky z nabídky nebo poklepání na ikonu. [9]

Mezi oblíbená GUI patří například GNOME nebo KDE. Obě prostředí jsou otevřeným softwarem, který si každý ještě může upravit dle svých potřeb. Mají také pracovní plochu, spodní panel s ikonami, mnoho aplikací, jako je prohlížeč, editor, poštovní software a spousty dalších. Samozřejmě nesou i nějaké rozdíly. V obou najdeme kreslící program, ale v GNOME je jiný než v KDE. Stejně tak se liší v textovém procesoru, jeden má KWord, druhý zase AbiWord. [11]

5.2 Příkazový řádek

V celé práci se ale s grafickým prostředím vůbec nesetkáme. Jediná část systému, která je pro nás důležitá, je příkazový řádek, respektive interpret příkazů. V něm budeme vypracovávat veškeré úlohy, které jsou ve sbírce úloh.

V následujících částech si představíme, s jakými příkazy přijdeme do styku a bude nutné vědět, k čemu je lze využít, jak je spojit nebo jaké argumentu u nich lze využít. Ze začátku je vybrán přehled několika základních a často používaných příkazů v úlohách, aby si žáci osahali prostředí, ve kterém se bude pracovat. Dále si ukážeme, jak se vytváří skripty přímo v Bashi a co do něj poté lze zapsat za příkazy pro řešení úloh.

5.2.1 Píšeme skripty v Bashi

Jsou chvíle, kdy je možné zapsat příkaz na jednu řádku a je zbytečné kvůli tomu vytvářet nový skript. Ovšem ve chvíli, kdy je potřeba provést nějakou složitější operaci, která se na jeden řádek nevejde nebo nějakou operaci musíme udělat několikrát dokola, pak stojí za úvahu, zda již není skript potřebný. V tuto chvíli je jisté, že to lze udělat i lépe, než pokaždé psát směs příkazů, aby systém něco vykonal. Je proto jednodušší tento kód zapsat do skriptu, který lze následně spustit kdykoliv a na jakémkoliv stroji. Jde vlastně o jakési zjednodušení a urychlení práce pro správce systému. [10]

Samotné skripty jsou vlastně textové soubory, které obsahují příkazy. Po spuštění skriptu systém začne vykonávat jeden příkaz za druhým tak, jak jdou po sobě a je ve skriptu nastaveno. Skriptem si můžeme velmi usnadnit práci, například spustit automatické zálohování počítače, či využít jednu možných funkcí, pro kterou byl zejména systém Linux vyvinut – zpracování textu. [9] Tedy například změnit či vyhledat nějaký text v souborech a následně ho změnit.

Skript v Bashi má určitou syntaxi, jakési pravidlo, které je vhodné dodržovat pro správnou funkci, aby se vyhnulo zbytečným možným problémům. Proto si v následující tabulce ukážeme, jak by měla vypadat hlavička skriptu.

```
#!/bin/Bash

# Název skriptu

# Popis skriptu
```

Tabulka 2 Hlavička skriptu v Bashi [9]

První tři řádku ve skriptu jsou uvedeny křížkem (# - hashem), tedy se neprovedou. Křížek při spuštění skriptu znamená, že se neprovádí. První řádek říká Linuxu, který program má použít k provedení tohoto skriptu. V tomto případě se jedná o tzv. shebang, který volá interpret Bash. Pokud bychom například psali skript v Perlu, první řádek skriptu by volal Perl. Další dva řádky jsou pouhými komentáři, který Bash ignoruje a přechází až k příkazům, které v komentáři nejsou. [11] Skript může obsahovat pouhý jeden příkaz, který by šel zapsat do příkazového řádku, ale také stovky dalších řádků s příkazy [14]. To však nabízí pouze skript, protože do příkazového řádku lze zapsat pouze jeden řádek příkazu. [9]

5.3 Příkazy pro univerzální použití

Do této kategorie bych rád zařadil příkazy, které je možné použít bez ohledu na prostředí, ve kterém se budou nacházet. Tedy není problém tyto příkazy použít v příkazovém řádku, ale můžeme je použít i ve skriptech.

5.3.1 Základní vybrané příkazy

Příkaz	**Co dělá**
Cat	Zobrazuje obsah souboru
Cd	Mění aktuální adresář
Cp	Kopíruje soubor nebo adresář
Date	Zobrazí aktuální datum a čas
Echo	Zobrazí text
Find	Hledá soubor na disku

Ftp	Přenáší soubory mezi počítači
Grep	Hledá soubor s určitým obsahem
Gropupadd	Přidá novou skupinu
Gzip	Komprimuje soubory
Chgrp	Mění skupinu souboru
Chmod	Mění přístupová práva souboru
Chown	Mění vlastníka souboru
Less	Zobrazuje obsah souboru
Ls	Zobrazuje všechny soubory v adresáři
Mkdir	Vytvoří adresář
Mv	Změní umístění souboru, ale může být použit i k přejmenování souboru
Passwd	Změní heslo účtu
Pwd	Ukazuje cestu k aktuálnímu adresáři
Rmdir	Odstraní adresář
Su	Přejde na účet superuživatele
Tar	Vkládá nebo vybírá soubory z archivů
Touch	Aktualizuje čas posledního přístupu
Useradd	Přidá nový uživatelský účet

Tabulka 3 vybrané příkazy [9]

V úvodu sbírky úloh nás budou zajímat zejména příkazy *echo*, *cat*, *date* a příkazy pro práci se soubory a složkami, jako například *mkdir*, *cp* či *mv*. V další části se s každým příkazem seznámíme trochu podrobněji, jaké má možnosti voleb.

5.3.2 Zobrazení textu (příkaz echo)

Tento příkaz slouží ke zobrazení textu nebo určené proměnné. Prvním příkladem nejčastěji bývá „Ahoj světe“. [9]

Příklad: echo „Ahoj světe“ [9]

U příkazu echo můžeme použít k formátu výstupu řadu speciálních znaků [9], které si představíme v následující tabulce.

Volba	Co dělá	Příklad (echo)
\a	Výzva (zvonek)	\achyba
\b	Backspace	\bahoj
\c	Potlačí nový řádek	$radek_textu\c
\f	Posun na novou stránku	Ahoj světe\f
\n	Začne nový řádek	Ahoj \nsvěte
\r	Návrat vozíku	Ahoj světe\r
\t	Vloží tabulátor	Ahoj \tsvěte
\v	Vloží vertikální tabulátor	\vahoj světe
\\	Znak zpětného lomítka	Zpětné lomítko \\
-e	Povoluje použití speciálních znaků	-e ahoj \tsvěte
-n	Nezačne za textem nový řádek	-n ahoj „světe“ Ahoj světe

Tabulka 4 volby příkazu echo [9]

5.3.3 Vytváření a mazání adresářů

Adresáře můžeme vytvářet a mazat v příkazovém řádku následujícími příkazy s určitými argumenty, díky kterým lze nastavit například smazání adresáře, který něco obsahuje. Bez přidání argumentu totiž není možné adresář odstranit z důvodu bezpečnosti. [9]

Příklad: mkdir skripty

Volba	Co dělá	Příklad (mkdir)
-p	Vytvoří rodičovské adresáře, pokud neexistují	-p /home/user/newdir
-m mod	Změní práva nového adresáře do zadaného módu	-m 777 dir24

Tabulka 5 volby příkazu mkdir [9]

Výše jsme si ukázali, jak se vytváří nový adresář. Níže si ukážeme, jak odstranit soubory či adresáře. K tomu lze využít příkazu *rm* či *rmdir* [9].

Příklad: rm skripty

Volba	Co dělá	Příklad (rm)
-d	Odstraní adresáře dokonce i když nejsou prázdné	-d /home/user/dir
-f	Odstraní bez varování i soubory bez práva zápisu	-f soubor.txt
-I	Varuje před smazáním souboru	-I test*.txt
-q	Uživatel není varován před smazáním	-q test.txt
-r	Smaže všechny soubory v adresáři i jeho podadresářích	-r /home/user/dir
-s	Odstraní zadaný adresář včetně všech podadresářů a souborů	-s /home/skripty
-v	Před smazáním zobrazí názvy souborů	-v zprava*

Tabulka 6 volby příkazu rm [9]

Výše zmiňovaný příkaz *rmdir* sám o sobě neodstraní adresář, který obsahuje nějaký soubor. Tedy je nutné k němu využít volbu *–r*. [9] Využito ve sbírce úloh v úvodních úlohách.

5.3.4 Kopírování, přejmenování a přesouvání souborů

Všechny zmíněné operace lze v příkazovém řádku provádět příkazy *cp* a *mv*. Záleží ovšem na volbě, která se k příkazu přidá. Pokud se nenastaví například volba *–i*, příkaz automaticky přepíše soubor bez varování.

Příklad zkopírování souboru: cp cesta/původnýjméno cesta/cílovéjméno

Příklad zkopírování adresáře: cp -r cesta/původnýjméno cesta/cílovéjméno

Příklad přesunu souboru či adresáře: mv cesta/původnýjméno cesta/cílovéjméno

Volba	Co dělá	Příklad
-f	Přepíše existující soubor	mv –f soubor /home/user
-v	Zobrazí jméno přesouvaného souboru	mv –v soubor1 soubor2
-i	Dá varování před přepsáním souboru	cp –i zprava1 zprava2
-v	Zobrazí název kopírovaného souboru	cp –v soubor1 soubor2

Tabulka 7 volby příkazu mv, cp [9]

5.3.5 Prohlížení a úpravy textových souborů

Textové soubory jsou ty, které obsahují pouze textové znaky bez jakýchkoliv informací o formátování. Tyto informace jsou uchovány v kódu ASCII, či jiném, kterému rozumí většina aplikací. [9] Těchto editorů textu existuje mnoho, my si však vybereme pouze některé.

Příklad prohlížení souboru v terminálu: cat soubor.txt

Příklad úpravy textového souboru: vi soubor.txt

Příkaz	Co dělá
cat soubor	Zobrazí obsah textového souboru
cat soubor1 soubor2 > soubor3	Spojí dva soubory do jednoho
cat > soubor_vystup	Vytvoří soubor s daným názvem, čeká na zápis textu, který bude následně uložen

Tabulka 8 volby příkazu cat [9]

5.3.6 Přesměrování výstupu

Jsou situace, kdy je nutné výsledek uložit například do textového souboru. Toho lze docílit přesměrováním výstupu za pomoci znaku špičaté závorky > (větší než). Tímto jedním znakem se zapíše výsledek do souboru. Pokud však chceme zapsat do již existujícího souboru s daty na jeho konec, pak je nutné zadat tyto znaky dva, tedy >>. [9]

Příklad zápisu do nového souboru: ls > adresar.txt

Příklad zápisu do existujícího souboru: ls >> adresar.txt

5.3.7 Příkaz grep

Pomocí tohoto příkazu jsme schopni naleznout soubory s určitým obsahem a zobrazit všechny řádky, ve kterých se vyskytne. [11]

Tvar příkazu grep: grep volby vzor soubory

Co se týká vzoru, ten určuje hledaný textový řetězec, kterým může být řetězec písmen, ale také regulérní výraz, stejně tak to může být jakékoliv písmeno. Soubory určují, kde bude hledaný řetězec vyhledáván. [14]

Příklad: grep „informatika“ *

Tento příkaz bude vyhledávat slovo *informatika* ve všech souborech v aktuálním adresáři a následně zobrazí všechny řádky, na kterých se řetězec vyskytuje. Můžeme však zadat cestu, kde bude řetězec vyhledáván, tedy například */home/user/vyuka*. [9]

Volba	Co děla
-c	Zobrazí počet nalezených řádků
-i	Při hledání shody ignoruje velká a malá písmena
-l	Zobrazí jména souborů, ve kterých nalezl shodu
-n	Vypisuje i čísla řádků se shodou
-r	Prohlíží i všechny podadresáře zadaného adresáře
-v	Vypisuje všechny řádky, ve kterých shoda nebyla nalezena
-x	Vypisuje pouze ty řádky, které se celé shodují s daným řetězcem

Tabulka 9 volby příkazu grep [9]

5.3.8 Příkaz find

Jelikož je Linux velmi dobře uzpůsoben pro práci s textem, není problém ani vyhledávání souborů. Jeho použití je velmi rozmanité a můžeme nastavit mnoho parametrů, které ovlivní konečný výsledek. Zapsaným příkazem *find* do příkazového řádku zavoláme funkci vyhledávání. [9]

Příklad: find cesta podmínky

Cestou je myšleno, kde budou soubory vyhledávány, kde můžeme použít tečku pro hledání v aktuálním adresáři, či lomítko pro hledání v celém souborovém systému. Lze použít i několik podmínek za sebou, aby bylo vyhledávání ještě více detailní. [10]

Volba	Co dělá	Příklad
-atime +n|-n|n	Čas posledního přístupu je více než | méně než | přesně n dní	find . –atime +5
-ctime +n|-n|n	Čas poslední změny je více než | méně než | přesně n dní	find . –ctime -5
-empty	Prázdné soubory	find . -empty
-group jméno	Soubory vlastněné určenou skupinou	find . –group user
-name jménosouboru	Vzorek hledaný ve jménu souboru	find . –name „test“
-print	Zobrazí výsledky na obrazovce ve výchozím nastavení	find . –name soubor –print
-size nc	Soubory dlouhé n znaků	find . –size 100c
-type d	Hledá soubory určeného typu, například d = adresář	find . –type d
-user jméno	Soubory vlastněné uživatelem	find . –user user

Tabulka 10 volby příkazu grep [9]

5.3.9 Zástupné znaky a uvozovky

Nejen skripty, které jsou určené pro příkazový interpret, mohou využít speciálních znaků, které mají určitý význam [15]. Následuje ukázka některých, které můžeme využít.

Znak	Co znamená
*	Zastupuje jakýkoliv řetězec v názvu
?	Zastupuje jediný znak v názvu
[…] např. [xyz]	Seznam nahrazení, který obsahuje soubor znaků, které se mají shodovat (souborx, soubory, souborz)
[!…] -> [xyz]	Seznam nahrazení, který obsahuje soubor znaků, který řetězec nemá obsahovat (soubora, souborb, nikliv soubox-z)
$	Uvozuje začátek názvu proměnné

Tabulka 11 zástupné znaky [9]

Dále nás také zajímají uvozovky, protože jich máme více druhů a každá varianta znamená pro interpret něco jiného. [9]

Příklad: echo $mesto „$mesto“ `$mesto`

Výsledek: Praha Praha $mesto

Zde lze sledovat, že jednoduché uvozovky vypnout speciální význam znaku dolaru a zobrazí doslovně všechny znaky uvnitř. [10]

5.4 Příkazy pro specifické použití

V této části si představíme příkazy, které je možné využít pouze při práci se skripty. Není totiž možné, abychom do jednoho řádku zapsali celý dlouhý kód, který je většinou složen z více částí.

5.4.1 Proměnné

Některá data potřebujeme uchovávat v takzvaných proměnných. Do nich data načteme a následně je můžeme zobrazit [9]. Jedná se tedy o jakési kontejnery, které jsou nějak pojmenovány a nesou informace, kterými může být například číslo nebo text. Proměnná může začínat téměř jakýmkoliv znakem, ale ne číslem. [10]

Příklad: předmět=informatika

Důležité je si uvědomit, že před znakem = nesmí být mezera, ale ani za ním. Při ukládání do proměnné není nutné používat uvozovky, pokud se jedná o jednoslovný řetězec. Jakmile je řetězec delší, například by se hledalo spojení informační technologie, pak již musí být v uvozovkách. [9]

Příklad: předmět="informační technologie"

Pro zobrazení obsahu proměnné využijeme zástupného znaku dolaru.

Příklad zápisu: $předmět

Příklad zobrazení: informační technologie

Příkaz nesoucí název *date* je pro nás ve sbírce úloh také důležitý, zejména pro vyřešení několika úloh. Díky této proměnné jsme schopni zobrazit datum, den i čas systému. Stejně tak jsme schopni zapsat datum do jiné proměnné. [9]

Příklad zápisu: echo date

Příklad zobrazení: pá dub 14 23:59:00

Zápis	**Výsledek**
+%m	Měsíc (4)
+%d	Den (14)
+%y	Rok (18)
+%D	04/14/18
+%Y	2018
+%A	Sobota
+%H	23
+%M	59 (pozor na záměnu s malým m)

Tabulka 12 volby příkazu date [12]

Ve sbírce úloh se setkáme s několika úlohami, ve kterých se přesně tato problematika řeší. Z tabulky lze tedy vyčíst, že nejvíce se dá pochybit ve chvíli, kdy se zamění velké a malé písmeno. Například rozdíl velkého a malého y není tak velký – pouhý delší a kratší formát zápisu. Pokud zaměníme malé a velké m, pak se místo měsíce zobrazí minuty.

5.4.2 Podmínky

Za pomocí příkazu *if* můžeme zajistit podmínku, která kontroluje například daný řetězec, zda je podmínka pravdivá. Můžeme stanovit, co se provede, pokud bude podmínka splněna, ale také když podmínka splněna nebude. [16] Existuje opět mnoho voleb, kterými můžeme podmínku ovlivnit při vyhodnocování. Lze testovat jak soubory, tak textové výrazy. [16]

Volba	**Co dělá**	**Příklad**
=	Text se rovná řetězci	$předmět = „informatika“
!=	Text se nerovná řetězci	$předmět != „informatika“
-eq	Rovná se celému číslu	$věk –eq 20
-gt	Číslo1 je větší než číslo2	$věk –gt 20
-ge	Číslo1 je větší nebo rovno číslo2	$věk –ge 20
-lt	Číslo1 je menší než číslo2	$věk –lt 20
-le	Číslo1 je menší nebo rovno číslo2	$věk –le 20
-ne	Nerovná se číslu	$věk –ne 20
-n	Řetězec je delší než 0	-n $předmět
-z	Prázdný řetězec	-z $předmět

Tabulka 13 volby pro testování textu [9]

V následující tabulce si vypíše příklady voleb, kterými můžeme ovlivnit výsledek testování souborů.

Volba	Co dělá	Příklad
-d	Soubor existuje a je to adresář	-d $vyuka
-f	Soubor existuje a je regulérní	-f $vyuka.txt
-nt	Soubor1 je novější než soubor2	Soubor1 –nt soubor2
-ot	Soubor1 je starší než soubor2	Soubor1 –ot soubor2
-r	Soubor existuje a lze jej číst	-r $vyuka.txt
-s	Soubor existuje a má délku větší než nula	-s soubor1
-w	Soubor existuje a lze do něj zapisovat	-w $vyuka.txt
-x	Soubor existuje a lze jej spustit	-x $vyuka.txt

Tabulka 14 volby pro testování souborů [9]

Podmínku samozřejmě můžeme složit z více voleb, aby byla přesnější. Proto máme možnost vybrat si z a*nd* nebo *or*. K tomuto lze využít volby *–a*, *-o*. [17]

Volba	Co dělá	Příklad
-a	Oba výrazy musí být pravdivé pro splnění podmínky	$věk –gt 30 –a $věk –le 50 (věk mezi 31 - 50)
-o	Alespoň jeden výraz musí být pravdivý pro splnění podmínky	$věk –gt 30 –o $věk –le 50 (věk > 30 nebo < 50)

Tabulka 15 volby operandů and, or [9]

Dále si ukážeme, jak se vlastní podmínka celá zapisuje. Je důležité dodržet určitou syntaxi, aby podmínka fungovala správně dle očekávání.

```
if [ podmínka ]
 then
          Příkazy
elif [ podmínka ]
     then
          Příkazy
else
          Příkazy
fi
```

Jak si můžeme všimnou, příkaz začíná *if* a končí *fi*. Pokud je podmínka splněna, po zápisu *then* se provedou příkazy. Další částí může být zanesená další podmínka, *elif*, ale není nutné. Následuje poslední část, kterou je *else*. Zde se provedou příkazy, pokud podmínka není splněna.

6 Způsoby ověřování

Úlohy byly postupně vytvářeny a vzápětí ověřované žáky ve výuce, zda splňují předem vytyčené cíle, předpoklady k řešení či například očekávání, jak bude výuka vypadat. Žáci z jedné třídy byli rozděleni do dvou skupin po patnácti a v obou skupinách se učila stejná látka. Rozdíl byl pouze v tom, že jeden týden byla první skupina s*kupina1* a další týden byla první skupinou ta druhá, tedy s*kupina2*.

Výuka probíhala zejména na základě vytvořeného tematického plánu na celý školní rok, kde nejprve žáci z první skupiny obdrželi úlohu, či sadu úloh, které museli vyřešit. Během této skupiny jsem jako pozorovatel výuky sledoval, co žákům dělalo problém, či co bylo naopak příliš jednoduché a nesplnilo tak mé očekávání.

6.1 Zavedení problémové výuky

Co se týče problémové výuky, která byla využita pro téma této diplomové práce, většina žáků totiž takovýto přístup neznala a bylo nutné je s ním předem seznámit, aby se tak naučili pracovat a sami chtěli další problémy řešit. Má příprava žáků na problémovou výuku byla taková, že jsme před Linuxem pracovali v prostředí Microsoft Windows, kde žáci pracovali v příkazovém řádku a předtím ještě v Microsoft Office Word a Excel. Ze začátku jsme řešili jednoduché úlohy společně a postupně byly úlohy ztěžovány. Žáci začali opravovat chybu v kódu u dávkových souborů a po čase dostali zadání problémem, který museli vyřešit. Jakmile jsme se dostali k výuce v Linuxu, žáci již chápali, co se od nich v zadání chce právě na základě zkušenosti z dávkových souborů.

Z tohoto důvodu pak samozřejmě záleží na učiteli, který by chtěl tuto sbírku využít, zda již žáky někdy touto metodou vyučoval či je to pro ně zatím neznámá. Pokud se s ní žáci doposud nesetkali, je vhodné, aby je na několika příkladech připravil na následující výuku v Bashi. S tím tedy souvisí, aby si učitelé své žáky připravili na jinou formu výuky a nebyla tak snížena kvalita potenciálu úloh.

6.2 Průběh pozorování

V této chvíli jsem v roli učitele působil jako „rádce“ pro nalezení správné cesty k dosažení cíle. Na začátku jsem zadal úkol a následně jsem očekával od žáků, že ho budou samostatně řešit. Pokud některému žákovi něco nefungovalo, individuálně jsem k němu přistoupil a snažil se mu oklikou poradit, co by měl udělat. Nikdy jsem

žákovi neřekl přímo, že například na řádku 10 má chybu v uvozovkách, ale že by si měl například v kódu zkontrolovat nějaké speciální znaky či symboly. Jakmile někteří žáci úlohu dokončili, jako učitel jsem jim zadal další práci k řešení. Úlohy pro druhou skupinu byly upraveny tak, aby již úloha / sada úloh splňovala co nejvíce očekávání. Výuka byla však postavená tak, že ne vždy první skupina prošla danou problematikou jako první. Vyučování na této škole je rozděleno na liché a sudé týdny, tedy jeden týden byla první skupina tou první, testovací, a druhý týden až tou druhou, kdy žáci dostali v případě potřeby úlohy upravené k lepšímu na základě pozorování a následného přepracování úloh, aby se co nejvíce přiblížily očekávanému cíli.

Během výuky jsem sledoval a komunikoval s žáky, jak daný příklad vnímají, zapisoval si postupně jejich dotazy a časy, v jakých úkol řeší. Na základě těchto poznámek jsem poté vyhodnocoval, zda daná úloha splnila mé očekávání a naučila žáky předem vytyčené cíle. Pokud tomu tak v nějaké části příkladu nebylo, například se žáci nemohli dobrat zdárného konce, bylo nutné upravit zadání příkladu či jim poskytnout část kódu pro zpřehlednění kódu a orientaci v něm.

Při výuce v první skupině jsem tedy s žáky vyzkoušel prvotní zadání a zapisoval žákovská řešení, jejich problémy při řešení či časy, za jak dlouho úlohu vyřeší například průměrní či nadaní žáci. Na základě těchto poznatků jsem poté mohl změnit zadání, pokud si to vzniklá situace vyžádala. Jakmile ke změně došlo, vyzkoušel se příklad s pozměněným zadáním ve skupině druhé a opět jsem zapisoval, jak žáci pracují, zda se blíží cílům lépe než ve skupině první.

Po otestování příkladu v obou skupinách, eventuálně s pozměněným zadáním, se úloha zařadila ve finální verzi do sbírky úloh s poznatky, jaká byla očekávání, a naopak, jak to vypadalo v reálné výuce či nějaké další poznámky.

Dále jsem během výuky zjistil, že pro žáky bylo výhodné, když si pro práci otevřeli dvě okna s terminálem. V jednom okně psali skript a ve druhém zkoušeli, jak funguje. Část žáků, kteří to dělali ve dvou oknech, byli o něco rychlejší v řešení, protože pouze klikali mezi okny, zatímco ti, kteří neustále otevírali a zavírali kvůli úpravě kódu a spouštění skriptu, byli tímto časově poněkud indisponováni.

Většina úloh je řešena za použití skriptů. Je tomu tak zejména z důvodu, aby žákům při výuce vznikaly rovnou výukové materiály. Pokud by vše zkoušeli pouze v terminálu, museli by následně příkazy přepisovat do svých poznámek, zatímco takhle si pouze uloží fungující skript, ze kterého se následně můžou učit.

7 Praktická část

V této části bych rád popsal, jakým způsobem byly úlohy vytvářeny, testovány ve třídách a na základě toho poté upravovány. U některých příkladů se totiž stalo, že očekávání se lišilo od reality. U některých příkladů se totiž očekávalo, že nad nimi žáci stráví pouze zlomek času, oproti tomu, kolik času na něm opravdu strávili. A zase naopak. Některé příklady byly dle předpokladů vymyšlené jako obtížné, ale žáci se s tím vypořádali mnohem rychleji. U každé úlohy je tedy proto napsáno, jaké bylo očekávání a jak to vypadalo během výuky, jaká byla realita.

Praktická část je rozdělena na dvě hlavní části. V první části se seznámíme se zadáním všech úloh ze sbírky, kde je rozebráno očekávání a realita daného úkolu. V části druhé jsou již metodické listy pro učitele, ve kterých je téma, obsah, zadání všech úkolů či například očekávané chyby v řešení úlohy. Součástí metodických listů jsou samozřejmě cíle výuky, předpoklady pro vyřešení a u některých příkladů i poznámky z hodin.

Dále bych rád upozornil, že v zadáních příkladů se můžeme setkat s texty, které jsou označené kurzívou. Toto nám značí, že se jedná o názvy skriptů, souborů, složek, eventuálně jsou to texty, které by se měly ve skriptech objevit.

7.1 Úloha 1: Přípravka

Tato úloha je první, se kterou se v Linuxu žáci setkají. Byly vybrány základní jednoduché příkazy, které mohou žáci použít přímo v terminálu, díky kterému rovnou uvidí výsledek, zda se jim operace zdařila či nikoliv. Tyto příklady jsou uvedeny v sadě úloh pod číslem 1, tedy přípravné úkoly. Je to například změna adresáře, výpis obsahu adresáře, tvorba, kopírování, přesouvání, mazání složek a souborů.

> Pomocí terminálu si vytvoř ve svém domovském adresáři novou složku, kterou pojmenuješ svým příjmením bez diakritiky. Následně do této složky vytvoř složku s*kripty* a do ní vytvoř soubor *den.txt*, do kterého zapiš aktuální datum a čas. Složku s*kripty* dvakrát zkopíruj a následně přejmenuj na *pracovni* a *pracovni2*. vypiš, jaké složky se aktuálně nachází ve tvém adresáři. Jakmile budeš mít zkopírováno, složku *pracovni2* smaž. Změň si u svého uživatelského účtu heslo a podívej se, jaké jsou v systému spuštěné procesy.

7.1.1 Očekávání

U této úlohy jsem předpokládal, že žákům zabere velmi krátký čas, protože se měli pouze dostat do svého domovského adresáře a tam vytvořit složku se svým příjmením, do ní složku s*kripty* a do této složky poté vytvořit soubor – *den.txt*.

Proto jsem od toho příkladu očekával, že žákům zabere několik minut vytvořit složky, protože se budou seznamovat s novými příkazy, i když byly podobné těm z příkazového řádku a již věděli, co mají hledat, ale žákům tato část paradoxně zabrala mnohonásobně delší dobu, než jsem očekával.

7.1.2 Realita

U této první části bych nečekal, že obě skupiny s ní již budou mít problém. Před problémovou výukou v Linuxu byli zvyklí na problémovou výuku ze systému Windows, jmenovitě z příkazového řádku a dávkových souborů, kde řešili také úlohy, které jim byly neznámé. Proto jsem byl zaskočen, že již tato první část vytváření souborů a složek v průměru žákům zabrala 50 minut. Nejrychlejší žák byl se vším hotový za 30 minut a nejpomalejším žákům, téměř polovina druhé skupiny třídy, toto vše trvalo 70 minut. Přes mé postupné postrkování, kdy jsem se taktně snažil naznačit, co by pravděpodobně měli hledat, aby se dobrali zdárného konce, tak postupně začali nalézat webové stránky, na kterých byla správná řešení.

7.1.3 Problémy při řešení

Někteří žáci tento úkol řešili velmi zdlouhavě, protože ze začátku vůbec nevěděli, co mají hledat. I když měli zadání česky, nebyli schopni ho přeložit do angličtiny, aby si nalezli to, co právě potřebují vyřešit. Po nějakém čase se jim však zadařilo. Největší problém pak nastal žákům ve chvíli, kdy měli za úkol smazat neprázdnou složku *pracovni2*. Ta totiž obsahovala soubor a nelze smazat neprázdný adresář bez zadání argumentu pro rekurzivní mazání bez oznámení.

7.2 Úloha 2: Uvítání do programu

Tato úloha má velmi nízké očekávání k dosavadním kompetencím žáků ohledně Linuxu. Předpokládá se, ho umí spustit a intuitivně v něm pracovat obdobně tak, jako jsou zvyklí z operačního systému Windows. Prvním požadavkem však je, aby nalezli terminál a v něm začali zkoušet příkazy, kterými se namapují do vytvářených složek a ve složkách poté vytvoří soubory.

Vytvoř si nový skript v Bashi a ulož ho na disk. Jakmile budeš mít vytvořený soubor, zapiš do něj skript tak, aby po jeho spuštění vypsal následující text:

„Toto je první řádek textu,

zde je druhý řádek výpisu, ale vše bylo původně zapsáno pouze v jednom řádku.“

Příkaz pro výpis zajistí vypsání obou řádků najednou. Tedy příkaz se v kódu objeví pouze jednou, nikoliv dvakrát. stejně tak zapíšeš text do jednoho řádku bez použití enteru. Pozor také na mezeru na začátku druhého řádku.

7.2.1 Očekávání

U této úlohy jsem předpokládal, že žákům již nějaký čas zabere, protože to již nebylo o pouhém zadání příkazu, ale o kombinaci příkazu a argumentu.

V samotném skriptování měli žáci vytvořit skript, který vypsal jednu větu a v půlce ji rozdělil na další řádek. Šlo zde o to, aby si vyzkoušeli příkaz echo a zjistili, že je možné zapsat i nějaký argument nejen na konec příkazu, ale i do jeho části. Což měl být problém, který měli žáci během chvíle vyřešit, protože prvním odkazem například v Googlu lze zadáním „bash echo new line“ naleznout stránku s vysvětlením a praktickým příkladem, jak lze použít. Proto jsem od toho příkladu očekával, že žákům tato úloha zabere pouze několik minut.

7.2.2 Realita

U této úlohy jsem byl poměrně zaskočen, že i když mohli žáci vyhledávat na internetu, tak jejich práce byla velmi pomalá. I nejrychlejšímu žákovi tato úloha trvala zhruba 15 minut, než ji dokončil. Očekával jsem, že je to otázka několika minut, respektive že do 5 minut by měli mít hotovo všichni žáci, protože stačí dobře formulovat vyhledávací dotaz a budou mít výsledek během chvíle.

7.2.3 Problémy při řešení

Některým žákům se nechtělo řešit zadání tak, jak bylo napsáno, a proto se ho snažili po nějakém čase obejít, aby to vypadlo, že mají splněno. Na první pohled vše vypadalo v pořádku, protože text byl rozdělen na dva řádky. Nicméně při roztažení okna terminálu se text postupně začal vracet do prvního řádku. Při kontrole kódu bylo zjištěno, že žák přidal takový počet mezer, aby mu to vyšlo přesně na druhý řádek při dané velikosti okna.

Dalšímu žákovi se stalo například to, že sice objevil, jak zapsat nový řádek, ale zaměnil lomítko za zpětné lomítko. V danou chvíli ve třídě nastala debata, jestli používá „to nko“, a pokud ano, tak mu to musí přece fungovat, protože to jinému žákovi tak funguje. Při jeho smůle se však žák s fungujícím kódem vyjádřil, že v kódu má písmeno n s lomítkem, nikoliv se zpětným lomítkem, tedy s tzv. backslashem.

7.3 Úloha 3: Kalendář

V této úloze by se žáci měli seznámit s tím, jak se pracuje s proměnnou systému *date* a jak je možné ji využít ve skriptování. Dále by pro ně měla být úloha přínosná také díky zjištění, že velikost písmen je při psaní v Bashi velmi důležitá.

> Vytvoř skript v Bashi, který vypíše čas v českém formátu, tedy hodiny:minuty, například 12:58. Skript dále vypíše datum v tzv. krátkém tvaru, tedy například 15.03.2018.

7.3.1 Očekávání

Naplánovaný čas pro tuto úlohu byl předem určen zhruba na 45 minut. Žáci měli sami naleznout na internetu možnost výpisu data a času a jeho možnosti formátování. Bylo zde na ně však připraveno i několik „pastí“, do kterých se mohli chytit. Zejména pak právě zmiňovaná velikost písmen, kdy při výpisu data je velmi rozdílné řešení pouhou změnou této velikosti. Zde jsem již očekával, že se žáci alespoň částečně sjednotí ve vyhledávání a řešení všem bude trvat obdobně dlouho.

7.3.2 Realita

Dle očekávání se žáci vešli s řešením do času od 20 minut do 40 minut. Také se sami dostali do problémů, se kterými jsem předem počítal a které museli vyřešit. Navíc si tam i sami vytvořili ještě další problém, se kterým jsem nepočítal, že by se tam mohl vyskytnout. S chybou v zápisu data jsem počítal, nicméně žák udělal chybu v zápisu času a nechápal, proč mu to vypisuje špatně čas. Tedy jak je více rozepsáno v problémech při řešení. Zde již žáci pracovali bez zásahu učitele. Jako bonus žáci ještě začali řešit příklad tak, že si v hlavičce definovali proměnné a ty následně vypisovali, aniž by používali „date“ s argumenty. To mne překvapilo asi nejvíce, protože další úloha je zaměřená na práci s podmínkami, se kterými se doposud setkali neměli. Realita však byla bohužel taková, že si našli kód na internetu, zkopírovali ho a začalo jim to částečně fungovat a už řešili pouze detaily. V první chvíli jsem očekával, že další úloha zaměřená právě na proměnné pro ně bude triviální, ale opak se stal pravdou.

7.3.3 Problémy při řešení

Žák se dostane do prvního problému ve chvíli, kdy zapíše špatně argumenty příkazu. Pokud nepoužije argumenty *„%H:%M“*, ale pouze jeden obecný argument „%T“, získá čas včetně sekund, nicméně hodiny v počítači či na hodinách zobrazují čas zpravidla pouze hodiny:minuty. Proto bude muset řešit, jak se zbavit sekund ve výpisu, kde dle předpokladu se bude snažit přidat nějaký další argument nebo nějak odmazat část znaků. Po jejich správném zadání či úvaze dojde žák k vyřešení problému tak, že změní argument, respektive vymění jeden za dva.

Pokud však žák použije u zápisu hodin „%H:%m“, setká se s problémem, že mu bude vypsána sice správně hodina, ale špatně minuta. Namísto minut se totiž zobrazí číslo aktuálního měsíce. Žák se tedy dostal do situace, kdy místo 10:45 se mu neustále zobrazovalo 10:03 a říkal: *„funguje to, ale nějak pofidérně, nechápu, proč je tam pořád ta trojka, i když už je v systémovém čase 10:50“*. Po chvíli hledání a testování chybu našel a opravil. Dále také může zapsat „%h:%m“, kdy se mu čas nezobrazí už vůbec, protože se vypíše zkratka a číslo měsíce.

Co se týká data, tak se žák dostane do dalšího problému při jeho zápisu, zejména u aktuálního roku, kdy při klasickém zápisu *„%d. %m. %y“* (dle očekávání fungujícího času *%h:%m*) se žákovi zobrazí *„15. 3. 18“*. Bude tedy muset řešit problém, proč se nezobrazí rok celý, ale ve zkráceném tvaru. Stejně tak se mu může zobrazit v delším tvaru den při zápisu z *„15. 3. 18“* na *„03/16/18. 03. 18“*, a to při zápisu kódu „%D.%m%y“. To vše pouze záleží, jakou velikost písmen žák při zápisu zvolí. Pokud bude částečně vycházet z času, kde použije zápis *„%H:%M“*, pak se mu zobrazí předchozí varianta. Kombinováním tedy postupně zjistí, že musí kód zformulovat tak, aby se zobrazil datum v používané formě, a to za použití *„%d.%m.%Y“*.

7.4 Úloha 4: Není rovná se jako rovnáse

Žáci se v této úloze měli seznámit s tím, jak je nutné zapisovat proměnné. V předchozí úloze zjistili, že důležitost použití velkých a malých písmen v Bashi je velmi vysoká. Stejně tak by díky této úloze měli zjistit, jakou roli hrají mezery. Dále by pro ně také mělo být významné zjištění, že posloupnost příkazů v kódu je velmi důležitá.

> Ve skriptu, který jsi právě obdržel, je chyba. Chceme, aby nám skript při spuštění napsal: *„Jaká barva vlasů je u holek nejhezčí? Možnosti jsou čtyři. Blond. Hnědá. Černá. zrzavá.“* Namísto toho skript píše chybu, že příkaz nebylo možné vyhledat a nezobrazí nám odpověď.

7.4.1 Očekávání

Očekávat lze, že někoho s předchozími zkušenostmi ze skriptování, například z dávkových souborů Windows, tento příklad vůbec nezaskočí a pouhým pohledem odebere mezery a vše začne fungovat. Nicméně i takto pochopí, jak proměnné fungují. Zjistí, že si musí dát pozor na mezery, kam je zapíše. Přiřazení hodnoty je funkční pouze bez mezery, tedy proměnné „barva“ lze přiřadit hodnotu „blond“ následovně: „barva=blond“.

V žákovských příkladech bude v kódu chyba právě v této mezeře (*barva = Blond; barva =Hnědá; barva = Černá; barva = zrzavá*).

Každá varianta bude odlišná pouze v tom, kde se mezera vyskytne a na základě toho by měl žák vyřešit problém, proč se proměnná nezobrazuje

Následně i po vyřešení tohoto problému se setkají s problémem druhým, že chyba přestane být vypisována, ale hodnota se ani tak nezobrazí kvůli posloupnosti příkazů v kódu. Hodnota proměnné se totiž přiřazuje až po výpisu této proměnné. Proto by si měli žáci uvědomit i to, že ve skriptování je posloupnost příkazů velmi důležitá.

Dále se také proměnná nastavuje záměrně až po příkazu echo, tedy žák i když zjistí, že je chyba v mezerách, hodnota proměnné se opět nezobrazí, i když už bez chyby příkazu. Záměrně je vynechán i znak dolaru, kterým se vypisuje obsah proměnné.

7.4.2 Realita

Většina žáků tuto úlohu řešila na základě zkušenosti s proměnnými z předchozí úlohy, nicméně i tak měli velký problém s předem připravenými mezerami. Nečekali,

že v mezerách bude takový problém a prvotně vůbec nehledali chybu zde, ale v názvech proměnných. Nicméně očekávaný čas, který měl být na 45 minut, byl poměrně vysoký, protože i nejpomalejší žáci tuto úlohu vyřešili do 35 minut.

7.4.3 Problémy při řešení

Jeden z problémů, se kterým se žáci potýkali už dříve, byl posun proměnných pod výpis jejich hodnot. To byl jeden z důvodů, proč žáci, i když vyřešili názvy proměnných, které byly špatně, tak jim stále byla vracena chyba a výsledek v nedohlednu. I přes tento neúspěch se nenechali odradit a hledali, kdy jinde by ještě mohla být problém. Postupně zkoušeli a odmazávali mezery, až zjistili, že je to opravdu už jen v mezerách, protože skript začal fungovat dle zadání.

7.4.3.1 Změny v zadání

Původně bylo zadání vymyšleno poměrně nezajímavou formou pro žáky. Ti měli dostat náhodně jeden ze tří skriptů a v každém byla pouze chyba v mezeře při přiřazování proměnné. Tedy ve skriptu byla proměnná pojmenována „A“ a hodnota nabývala hodnoty „1“ (A=1; A =1; A= 1). Toto však není příliš motivující zadání a žáci by si řekli, že to dělají, jen protože to učitel chce, nikoliv, že mají sami něco objevovat a mít zájem problém vyřešit. S postupnou změnou tohoto zadání se dospělo ještě k dalším změnám Jednou z nich byl přesun proměnných až na konec kódu, aby si žáci uvědomili, že je zde nutné definovat nejprve hodnotu proměnné a až poté je možné s ní pracovat, respektive vypsat její hodnotu. Další změnou pak byla už pouze záměrná syntaktická chyba, a to odebrání znaku dolaru z výpisu hodnoty proměnné. Tedy z pouhé úlohy, kterou bylo možné vyřešit i pouhým logickým uvažováním, se stala úloha mnohem složitější s rozmanitějším charakterem problému.

Ještě jsme konzultovali jednu možnost, a to omezit zápis proměnné pouze na jednu variantu, například „A = 1“. Zejména kvůli povaze úlohy, aby každý z žáků měl stejnou možnost úlohu vyřešit a některý nebyl zvýhodněn zadáním. Z tohoto důvodu se změnilo téměř celé zadání včetně předaného skriptu žákům.

7.5 Úloha 5: Podivuhodná kalkulačka

Tato úloha je částečně zaměřená na zopakování a prohloubení získané znalosti z předchozího příkladu, tedy jak se zapisují proměnné, jak musí být umístěné či jaké hodnoty mohou obsahovat. Dalším důležitým mezníkem této úlohy je zápis aritmetických operací, kdy se žáci musí naučit, jak lze pomocí Bashe počítat.

> Ve skriptu *kalkulacka.sh* zjisti, proč ani jedna operace neudělá dle komentáře to, co by dělat měla. Nezapomeň na začátku upravit syntaxi zápisu pro výpočet aritmetických operací. Zjisti, kde je v prvních úlohách problém, že ani jedna operace nefunguje a jakmile to zjistíš, zaměř se na předposlední příklad, na dělení. Vyjde výsledek při tomto zápisu přesně tak, jak očekáváš?

7.5.1 Očekávání

V této úloze se od žáků očekává, že se naučí správný zápis pro aritmetické operace. Časově by tato úloha měla být náročnější než ta předchozí, na které se žáci měli pouze naučit, jak dosadit do proměnné hodnotu. Zde již tuto znalost musí ovládat, a navíc zde budou muset vyřešit ještě problém s aritmetikou. Časová dotace byla počítána maximálně na 45 minut, tedy na jednu vyučovací hodinu.

7.5.2 Realita

Většina žáků pracovala dle očekávání, ale někteří z nich samozřejmě byli pomalejší a někteří rychlejší. Ti, kteří zvládli vyřešit proměnné již v předchozí úloze, neměli moc velký problém zjistit, kde se chyba v zápisu proměnné nachází a poměrně rychle ji opravili. Po chvíli zjistili, že jsou špatně zapsané i názvy proměnných, ale ani tak jim to stále nefunguje. Po dalším bádání se žáci dobrali výsledku, že jsou ještě špatně umístěné proměnné. Nejrychlejšímu žákovi toto zabralo 25 minut a nejpomalejšímu tato úloha zabrala něco málo přes 60 minut. Nicméně valná většina žáků použila proměnné již při práci s datem a časem, následně při objevování přiřazování hodnot proměnným a celkový zápis proměnných, což bylo hlavním tématem čtvrtého příkladu. V tomto příkladu již žáci, kteří použili proměnnou u data a času, měli mnohem jednodušší práci v porovnání s žáky, kteří tuto proměnnou použili až v předchozím příkladu.

7.5.3 Problémy při řešení

Největším problémem při řešení této úlohy bylo modulo. Žáci se doposud nikdy nesetkali s takovýmto způsobem „dělení“. Často píší různé znaky právě pro dělení, takže si většina z nich ani neuvědomila, že znak procenta by v této úloze mohl špatně fungovat a celý výpočet kazit. Hledali chybu v syntaxi, porovnávali s fungujícím sčítáním, odečítáním a násobením, ale dělení stále nefungovalo. Až ve chvíli, kdy se nad jedním počítačem sešlo více žáků, teprve poté se jim podařilo úlohu vyřešit. Stalo se takto v obou skupinách. Mimo jiné žáci také měli problém se zápisem pro aritmetické operace, kde všude musí použít uvozovky, kde musí použít znak dolaru a jak příklady obalit do závorek. Nicméně to byl jeden z normálních problémů, který nebylo až tak složitě vyřešit.

7.5.4 Změny v zadání

Celá úloha byla inspirována výukou z dávkových souborů, kde již tito žáci museli naprogramovat „kalkulačku“. Na základě této jejich zkušenosti jsem vycházel, že by mohli být schopni řešit něco obdobného v Bashi. Proto jsem připravil úlohu s poměrně jednoduchým a striktním zadáním, které sice problémovou úlohu připomínalo, nicméně nebyla tolik promyšlená. Žákům stačilo, aby na internetu nalezli, jakou syntaxi má zápis aritmetických operací a poté již všechny řádky udělali správně. Postupně tato úloha přibírala na složitosti a propracovanosti. Změnily se názvy proměnných z anglických názvů na české, změnila se znaménka, z „-„ se stalo „mínus“, z „+“ se stalo „plus“ a podobně.

Byla tak žákům odebrána jakási forma pomoci, jak by zápis mohl vypadat. Poslední změnou bylo, že se zápis kódu zjednodušil. Na každé řádce v jednom příkladu se řeší pouze jedna věc. V prvním je výpis příkladu, který se bude počítat. Ve druhém řádku je zapsán aritmetický výpočet. A ve třetím řádku je vypsán výsledek. V první verzi této úlohy bylo vše zapsáno v jednom řádku, což nebylo dostatečně přehledné a mohlo být pro některé studenty matoucí, která část je zrovna početní a která pouze vypisuje data.

7.6 Úloha 6: Je dopoledne nebo odpoledne?

Tato úloha je záměrně zjednodušena pro začátečníky, kteří se ještě nikdy nesetkali s programováním v Bashi. Jakmile začnou řešit tuto úlohu, zjistí, že budou muset vyhledat příkaz, který rozhodne, zda je podmínka splněna či nikoliv. Pro tuto úlohu postačí pouhý jednoduchý podmíněný příkaz.

> Vytvoř si nový skript *castDne.sh* a vymysli ho tak, aby po spuštění vypsal, zda je před polednem či po poledni.

7.6.1 Očekávání

Časově na tuto úlohu bylo počítáno kolem 20 – 30 minut. Žáci po přečtení zadání měli začít vyhledávat na internetu, jak se podmínka zapisuje, aby mohli tento jednoduchý problém vyřešit.

7.6.2 Realita

Žáci po přečtení zadání začali se zápisem času, aby ho mohli porovnávat následně v podmínce. Velká část z nich už však zapomněla, jak se vůbec dá zapsat čas do proměnné, aby s ním mohli následně pracovat. Tím začalo první vyhledávání, aby mohli pracovat s časem. Jakmile čas našli, zapsali si ho tak, jak v úvodním příkladu, tedy „hodiny:minuty“, aby viděli, kolik je hodin.

Další částí této úlohy bylo vyhledat, jak zapsat do kódu podmínku *if*. Po určitém čase, v průměru 15 minut, měli téměř všichni žáci nalezenou podmínku, kterou zapsali do kódu a poté mohli začít upravovat dle zadání pro čas.

Největším problémem této úlohy však bylo porovnávání času v podmínce s hodinami, které jsou načtené ze systému. Tedy žáci měli porovnat, zda je aktuálně více či méně hodin než 12:00. Dle toho určovali dopoledne či odpoledne. Někteří žáci si uvědomili, že bylo možné v dávkových souborech zapsat čas bez dvojtečky, aby ho bylo možné porovnat. Než se toto vše žákům podařilo, uběhlo celkem 90 minut času.

7.6.3 Problémy při řešení

První problém této úlohy byl, že žáci zapomněli, jak se zapisuje čas do proměnné. Jakmile toto žáci vyřešili, museli vyřešit o mnoho větší problém. A to porovnání hodin z proměnné s hodinami, které se zapíší do kódu, například 12:00. Pokud žáci zapsali čas v takovéto formě, podmínka nefungovala dle očekávání, Bash vypsal chybu a podmínka nebyla splněna dle očekávání.

7.6.4 Změny v zadání

Tato úloha byla prvotně vymyšlena tak, aby na ní mohli žáci pracovat delší dobu, tedy například úloha v celkovém čase na čtyři vyučovací hodiny. Je v ní spojeno téměř vše, co se doposud žáci měli naučit, kromě podmínek. Nicméně bylo rozumnější jednu úlohu rozdělit na tři a žáky postupně učit od jednodušších podmínek, až po ty složitější. Proto se v první úloze setkáme pouze se zápisem podmínky „if-else“, aby si žák dokázal vyzkoušet na jednoduchém příkladu, jak podmínky fungují. V následujících příkladech se podmínky ztíží, protože žák bude muset do podmínky vnořit další podmínku do podmínky, složitější porovnávání a kombinace operandů s časy.

7.7 Úloha 7: Je dopoledne, odpoledne nebo poledne?

Úloha inspirovaná tou předchozí, kde navíc nabyla na složitosti přidáním dalšího časového údaje, tedy poledne. Do jedné podmínky „if-else“, která stačila v předchozím případě, je nutné vnořit ještě jednu podmínku, která bude zajišťovat právě kontrolu poledne.

Vytvoř si nový skript *castDne2.sh* a vymysli ho tak, aby po spuštění vypsal, zda je před polednem, po poledni či je právě poledne. Nezapomeň, že poledne je jen určitou chvíli.

7.7.1 Očekávání

V této úloze se od žáků očekává, že se naučí vnořit podmínku do podmínky, tedy zjistit, zda je dopoledne či odpoledne, eventuálně je přesně 12:00, a je tedy poledne. Časová dotace byla pro průměrného žáka počítána na 30 minut. U žáků bylo předpokládáno, že budou mít nejvíce problém s vnořením podmínky. Ostatní ve výsledku zůstane stejné, jako tomu bylo v předchozí úloze.

7.7.2 Realita

Tento příklad vyšel téměř dle očekávání. Jak časově, tak i s problémy, které mohly vzniknout při jeho řešení.

7.7.3 Problémy při řešení

Žáci řešili problém s vnořením podmínky. Jakmile se jim na internetu podařilo nalezнout, jakým způsobem je nutné do kódu tuto podmínku rozepsat, nevznikal v danou chvíli žádný další jiný velký problém. Pouze jeden menší, avšak velmi zajímavý. Žák se zeptal, jak má ozkoušet, jestli mu funguje poledne, když je zrovna 9:50 hodin. Po chvíli přemýšlení se zasmál a sám změnil systémový čas v počítači, aby mohl ozkoušet, zda mu toto vše funguje. Dalším řešení bylo, že žák do podmínky, kdy se kontrolovalo poledne, vepsal aktuální čas namísto času poledního. Tím byla podmínka splněna a fungovala.

7.8 Úloha 8: Jaká je část dne?

V této úloze si už žáci budou muset vyzkoušet kombinování podmínek s více argumenty, aby byla zajištěna jejich správnost. Budou totiž muset zjistit, zda se jedná o: noc, dopoledne, poledne, odpoledne nebo večer. Díky přidáním dalších části dne se podmínky rozrostly mnohem více, než tomu bylo doposud.

> Vytvoř si nový skript *castDne.sh* a vymysli ho tak, aby po spuštění vypsal, zda je noc, dopoledne, poledne, odpoledne, večer či půlnoc. Nezapomeň, že poledne a půlnoc je jen určitou chvíli. Dopoledne, odpoledne a večer si rozvrhni dle znalostí, v jakých časech se tyto části dne nachází.

7.8.1 Očekávání

U této úlohy se předpokládá, že žáci budou čerpat nabyté znalosti z předchozích příkladů, jak už ze zápisu data a času, proměnných, tak ze zápisu podmínek. Bylo zde počítáno, že žáci budou tuto úlohu dělat o mnoho delší dobu, než všechny doposud řešené úlohy. Proto byla časová dotace zvolena na 90 minut. Dalo se očekávat, že žáci budou mít problém s tím, že budou muset řešit čas ne pouze jedním porovnání, tedy například je více než 12:00, ale i zda je méně než 18:00)

7.8.2 Realita

Dle očekávání se i tento příklad povedl stihnout téměř všem v časové dotaci, jen někteří pomalejší to dodělávali ještě během další hodiny. Nejrychlejší žáci však tuto úlohu řešili zhruba 60 – 70 minut, což i tak splňuje očekávání.

Co se týká problémů, ty byly naplněny také dle předpokládaných možností.

7.8.3 Problémy při řešení

Žáci měli největší problém ve chvíli, kdy museli vnořit několik podmínek do sebe. V danou chvíli řešili, jak moc podmínky vlastně musí kombinovat, jaké časy vybrat, jaké argumenty pro správnost použít.

7.9 Úloha 9: Výběr skladeb oblíbeného interpreta

Toto je úvodní úloha pro práci s textem v textových souborech. Žáci se postupně budou učit, jak se s příkazy pracuje. V tomto příkladu se žáci naučí vypsat seznam na obrazovku. Seznam skladeb čítá 14100 řádků, tedy žák by měl být motivován k tomu, aby skript vytvořil a nehledal výsledky ručně.

Vytvoř si nový skript *interpret.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a následně vypsal veškeré jeho skladby ze seznamu do okna terminálu. Zkontroluj, zda jsi vybral opravdu všechny.

7.9.1 Očekávání

Od žáků se očekávalo, že budou mít problém s vypsáním všech řádků daného interpreta skriptem, protože jméno interpreta se musí zadat po spuštění skriptu z klávesnice. Tedy po každém spuštění by se skript měl dotázat, jakého interpreta má vyhledat. Na úlohu bylo počítáno kolem 15 – 20 minut.

7.9.2 Realita

Žáci neměli problém s nalezením příkazu, jak zadat do proměnné vstup z klávesnice, což bylo výhodou, že se nezdržovali a mohli se rovnou soustředit na vymýšlení příkazu, který zobrazí interprety na obrazovku. Někteří žáci si prvně psali příkaz přímo v okně terminálu bez zápisu a spouštění skriptu. Ve chvíli, kdy jim začal příkaz zobrazovat požadované řádky, přesunuli se k tvorbě skriptu a zadávání vstupu z klávesnice.

7.9.3 Problémy při řešení

Největší problémem pro žáky bylo naleznout příkaz, kterým se vyhledává text v souborech. Toto však po nějakém uplynulém čase žáci zvládli a postoupili v řešení příkladu.

Dalším problémem bylo, kdy měli žáci zapsat správně do apostrofů proměnnou, jejíž obsah zadávali z klávesnice. Do chvíle, než apostrofy použili správně, dostávali nejisté či špatné výsledky z vyhledávání.

7.10 Úloha 10: Výběr skladeb oblíbeného interpreta s pojistkou

Tato úloha je volným navázáním na úlohu předchozí, kde žáci mají pouze skript upravit tak, aby měli jistotu, že vždy vyberou všechny skladby interpreta, i když by se v seznamu jméno interpreta vyskytlo v jiné velikosti písmen, tedy, že by například nebylo první písmeno velké.

> Vytvoř si nový skript *interpretP.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a následně vypsal veškeré jeho skladby ze seznamu. Zkontroluj, zda jsi vybral opravdu všechny.

7.10.1 Očekávání

Očekávalo se, že žáci pouze vyhledají argument, který přiřadí k již fungujícímu příkazu a nezabere jim to dlouhou dobu.

7.10.2 Realita

Většina žáků během chvíle úkol vyřešila bez menších problémů dle očekávání.

7.10.3 Problémy při řešení

Problém se vyskytl pouze u několika žáků, kteří nepochopili, co se od nich požaduje. Učitel by v tuto chvíli měl zakročit tak, že žáky navede, s čím by se skript mohl v původní podobě setkat a co vybírá. Pokud žáci chápou, co za skript vytvořili, pak logicky sami pochopí, že příkaz vybere pouze přesně zadanou syntaxi. Pokud ani tak ne, učitel může žákům napovědět, že se jedná o velikost písmen při vyhledávání.

7.11 Úloha 11: Skladby oblíbeného interpreta do souboru

Tato úloha je dalším pokračováním předchozích úloh. Jediný rozdíl je zde takový, že se vyhledané řádky nezobrazí na okně obrazovky, ale zapíší se do nového textového souboru.

> V této úloze budeš muset naprogramovat skript tak, aby bylo možné zadat jméno oblíbeného interpreta a skript zapsal veškeré skladby do nově vytvořeného textového souboru.

7.11.1 Očekávání

Od žáků se očekávalo, že budou muset zjistit, jak je možné přesměrovat výstup vyhledaného výsledku. Úloha by neměla být pro žáky nijak zvlášť složitá, protože jde opět o rozšíření úloh předešlých, kde základ zůstává stejný. Časově jsem očekával od žáků vyřešení do 15 minut.

7.11.2 Realita

I nejpomalejší žáci tuto úlohu měli vyřešenou do 10 minut, tedy byli rychlejší, než jsem od nich očekával.

7.11.3 Problémy při řešení

Problémy se v tomto příkladu nevyskytly mezi žáky žádné.

7.12 Úloha 12: Počet skladeb oblíbeného interpreta

V této úloze jde o rozšíření předchozích úloh, kde však vyhledáváme stále řádky obsahující daného interpreta, ale nezajímají nás názvy, nýbrž jejich počet. Daný počet řádků se poté zapíše do nového textového souboru.

> Vytvoř si nový skript *interpretPocet.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a následně zapsal počet skladeb ze seznamu do textového souboru ze seznamu.

7.12.1 Očekávání

V této úloze jsem od žáků čekal, že budou mít největší problém se zjištěním, že se budou muset seznámit sami s fungováním roury v Linuxu. Příkaz *wc* jsem očekával, že naleznou poměrně rychle, ale budou muset vymyslet, jak spojit předchozí fungující kód s novým. Časovou dotaci jsem předpokládal na 30 minut.

7.12.2 Realita

Realita tohoto příkladu byla velmi zajímavá. Někteří žáci řešili příklad dle očekávání, tedy k původnímu kódu se snažili přidat rouru a *wc*. Někteří žáci však nalezli, že u příkazu *grep* máme i argument, který umí spočítat počet řádků. Někteří toho využili a vyhnuli se tak rouře a *wc*. Nicméně za toto byli pochváleni, že našli další možnost a vyzváni, ať se pokusí naleznout nějakou jinou možnost.

7.12.3 Problémy při řešení

Největším problémem u tohoto příkladu bylo spojení původního a nového kódu. Žáci nevěděli, jak tyto dva kódy spojit a když už se dohledali způsobu, že se to dělá pomocí roury, tak narazili na další „velký“ problém. Jak napsat znak pro rouru. Jakmile znak nějakým způsobem zapsali, mohli vyzkoušet, zda kombinace příkazů dělá to, co dělat má a měli vyhráno.

7.13 Úloha 13: Počet skladeb oblíbeného interpreta v terminálu

V této úloze se žáci naučí kombinovat příkazy za sebou. Tedy spojit příkazy, které již znají, tak, aby se dobrali požadovaného výsledku.

> Vytvoř si nový skript *interpretPocetT.sh*, který spočítá počet skladeb. Vymysli ho tak, aby se po spuštění zeptal na jméno interpreta, uložil výsledek do textového souboru a následně obsah souboru vypsal do okna terminálu. Pro spočítání ale nepoužívej k vyhledávacímu příkazu žádný argument, ale kombinaci několika příkazů.

7.13.1 Očekávání

Na tuto úlohu bylo počítáno zhruba 15 minut, protože již žáci měli kód připravený z předchozích příkladů a věděli, jak spočítat řádky, i jak je uložit do soubor, stejně tak věděli, jak zobrazit obsah souboru. Problémem v této úloze však pro ně bylo, že museli všechny tyto příkazy spojit do jednoho za pomoci výpisu posloupnosti provedených příkazů.

7.13.2 Realita

U žáků byl tento příklad trochu složitější, než se očekávalo. Většina neměla problém s tím, aby uložili výsledek vyhledávání do souboru, ani to, aby následně samotný výsledek příkazem zobrazili. Problém však nastal v tom, kdy měli uložit a zároveň ihned zobrazit. Někteří z nich se pokoušeli řešit i například pomocí roury, ale bezvýsledně. Nejrychlejší žáci tento příklad vyřešili za avizovaných 15 minut, nicméně většině žáků tento příklad zabrat zhruba půl hodiny.

7.13.3 Problémy při řešení

Žáci řešili, jak spojit příkazy tak, aby bylo možné data vyhledat, uložit a následně rovnou zobrazit. Toto vše samostatně neměli problém udělat, nicméně po uložení zobrazit počet řádků již problém byl. Příkaz pro zobrazení obsahu textového souboru měli v první úloze a delší dobu přemýšleli, jakým příkazem data pouze zobrazit. Soubor totiž otevírali v textovém editoru, tedy sice zobrazili výsledek, ale ručně, a ještě ne přímo výpisem do terminálu. Jakmile si vzpomněli či vyhledali příkaz pro zobrazení obsahu souboru, *cat*, začali žáci vyhledávat, jak spojit příkazy tak, aby se po uložení do souboru ihned výsledek zobrazil v terminálu.

7.14 Úloha 14: Výběr skladeb druhého oblíbeného interpreta

Tato úloha je navázáním na úlohu, ve které se žáci naučili přesměrovat výstup do textového souboru. V této úloze se totiž jedná o obdobný problém, kde žáci musí připsat výsledky do již vytvořeného souboru, aniž by přepsali obsah souboru.

> Vytvoř si nový skript *druhyInterpret.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a následně připsal veškeré jeho skladby ze seznamu na konec textového souboru skladby.txt. Soubor by měl po zápisu obsahovat skladby obou interpretů.

7.14.1 Očekávání

V této úloze se žáci měli zaměřit na již hotový skript, který pouze upraví tak, aby se nalezené výsledky zapsaly do již existujícího souboru, ve kterém již nějaké výsledky uložené jsou. Dalo se očekávat, že někteří žáci obsah souboru přepíší, namísto toho, aby tam data připsali. Na úlohu bylo počítáno 15 minut.

7.14.2 Realita

Časová dotace na tento příklad byla dle očekávání pro všechny žáky dostatečná, protože většina žáků bez větších problémů nalezla řešení a skript upravili dle požadavků. U některých žáků se však potvrdilo to, že si přepsali obsah celého souboru, i několikrát po sobě, tedy nepřepsali skript správně, aby data připisoval na konec souboru.

7.14.3 Problémy při řešení

Problémem u tohoto příkladu bylo pouze to, že někteří žáci ve skriptu špatně změnili skript, který namísto připisování dat data přepisoval. S ničím jiným se žáci zde nesetkali.

7.15 Úloha 15: Skladby oblíbeného interpreta do textového dokumentu i do terminálu

Žáci by v této úloze měli skript napsat tak, aby udělal ve výsledku to stejné, co skript udělal v předchozí úloze, ve které se nalezené výsledky uloží do souboru a následně se obsah souboru zobrazí v okně terminálu. Rozdílem však je, že se nezobrazí počet, ale obsah celých řádků.

> Vytvoř si nový skript *interpretTerminal.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a vytvořil textový dokument, do kterého zapíše názvy všech jeho skladeb a zároveň se nalezené skladby zobrazí na obrazovce terminálu.

7.15.1 Očekávání

U tohoto příkladu bylo očekáváno, že žáci pouze prohloubí svou znalost z předchozího příkladu, kde již téměř stejnou věc udělali. Pouze namísto toho, aby zapsali počet řádků dle vyhledávání do souboru, tak zapíší obsah celých řádků, tedy skladby vyhledávaného interpreta. Časově tato úloha měla být na několik minut.

7.15.2 Realita

Žáci začali tento úkol řešit tak, že si vzali svůj původní skript a snažili se ho přepsat na požadované změny. Některým se to povedlo opravdu během několika minut, někteří s tím měli problém trochu větší. Nejpomalejšímu žákovi trvalo tento úkol vyřešit 20 minut.

7.15.3 Problémy při řešení

Několik žáků ponechalo skript v původní podobě i s příkazem *cat* a snažili se do toho ještě přidat další příkazy či rouru. Postupně však zjistili, že tudy cesta nevede a musí skript pozměnit trochu jinak. A to tak, že vlastně stačilo odebrat ze skriptu právě příkaz *cat*.

7.16 Úloha 16: Skladby oblíbeného interpreta do textového dokumentu i do terminálu 2

Žáci mají za úkol, aby napsali skript, který dělá stejnou věc, jako již skript dříve vytvořený. Rozdíl je však v použití příkazů, kterým dosáhnou výsledku. Tedy naleznout skladby interpreta, zapsat je do souboru a obsah tohoto souboru následně zobrazit v okně terminálu.

> Vytvoř si nový skript *interpretTerminal2.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a vytvořil textový dokument, do kterého zapíše názvy všech jeho skladeb a zároveň se nalezené skladby zobrazí na obrazovce terminálu. Nepoužívej však kombinaci příkazů jako v předešlé úloze.

7.16.1 Očekávání

U žáků bylo očekáváno, že budou vyhledávat, jakou další kombinací příkazů dosáhnou požadovaného zápisu do souboru a zobrazení do terminálu. Problém však nebyl brán jako velký, protože stačilo vhodně zformulovat klíčová slova pro vyhledávač a po pročtení několika stránek bylo možné naleznout obdobné příkazy, které žáci mohli částečně použít, kde stačilo udělat pouze několik úprav.

7.16.2 Realita

Realita v tuto chvíli byla téměř totožná s očekáváním. Nejlepší žáci ve třídě byli s tímto skriptem hotovi během několika minut. Průměrným žákům tento příklad zabral o něco déle, ale v porovnání s nejpomalejšími žáky to až takový rozdíl nebyl. Časově rozpětí bylo 7 – 16 minut.

7.16.3 Problémy při řešení

Problémem u některých žáků bylo, že neuměli správně formulovat problém pro vyhledávač, tudíž nebyli schopni naleznout správné řešení. Postupně za pomoci překladače z češtiny do angličtiny vymysleli klíčová slova tak, že se jim nakonec podařilo dobrat ke končenému výsledku a příkaz *tee* naleznout.

7.17 Úloha 17: Kdo hledá, najde

V poslední úloze této sbírky měli žáci za úkol napsat takový skript, ve kterém se naučí, jak se vyhledává v názvech souborů. Aby se u žáků vyloučilo vyhledávání těchto souborů ručně, je pouze ve třech souborech z deseti tisíc obsah, ostatní jsou prázdné. Navíc názvy jsou náhodně generované a poměrně dlouhé.

> Vytvoř si nový skript *hledejSmysl.sh* a vymysli ho tak, aby ze všech souborů ze složky *smes* vybral pouze ty soubory, které obsahují v názvu souboru text *vtip*. Jakmile je najdeš, zobraz si jejich obsah.

7.17.1 Očekávání

V tomto příkladu se u žáků očekává, že se budou opět muset sami seznámit s příkazem *find*, kterému musí přiradit několik argumentů pro přesné definování vyhledávaných souborů. Tedy nastavit, kde se mají soubory vyhledávat a zároveň, jaký název soubory mají mít. Náročností je však tento úkol poněkud složitější, proto bylo na tuto úlohu věnováno 30 minut.

7.17.2 Realita

Žáci se ještě s tímto příkazem nesetkali, proto všichni začali vyhledávat, jakým příkazem je možné vyhledávat v názvech souborů. Jakmile se žákům podařilo naleznout, že se jedná o příkaz *find*, pak již během chvíle nalezli, jak zadat cestu, ve které se má vyhledávat a následně i formu pro název souboru, aby skript vyhledával správně.

7.17.3 Problémy při řešení

U této úlohy vzniklo problémů hned několik. Úvodním problémem byla pouhá neznalost tohoto příkazu a opět neschopnost formulovat správně text pro vyhledávač. Dalším problémem, se kterým se žáci museli vypořádat, bylo zadání cesty pro vyhledávání. Skript totiž měli uložený jinde, než měli uloženou složku se soubory, ve kterých se mělo vyhledávat. Tudíž měli problém se správným zadáním cesty, aby skript vyhledával pouze v zadané složce. Posledním problémem, avšak tím největším, byl pro žáky název souborů. Text *vtip* se totiž nachází náhodně uvnitř celého názvu, tedy není ani na začátku, ani na konci, ale někde náhodně uprostřed. Tam však nastal problém, protože žáci museli naleznout, jaké zadat zástupné znaky v názvu, aby skript vy-

hledával v názvech tak, že vyhledávaný text může být kdekoliv v celém názvu. Zobrazovaly se jim nejprve například soubory všechny nebo naopak zase žádné soubory. Dále se žákům také zobrazil text z úvodu skriptu. Nakonec se jim však povedlo zadat zástupné znaky tak, aby se zobrazilo vše tak, jak bylo zadáno.

8 Sada problémových úloh

8.1 Úloha 1

Téma: Přípravné úlohy

Název úkolu

Přípravka

Obsah

V této úloze budeš muset zjistit, jak se pracuje se soubory a složkami. Tedy například vytvořit, zkopírovat či smazat složku, zobrazit si obsah adresáře a textového souboru. Poté také vyzkoušíš změnit heslo uživatele a vypsat si spuštěné procesy.

Zadání

Pomocí terminálu si vytvoř ve svém domovském adresáři novou složku, kterou pojmenuješ svým příjmením bez diakritiky. Následně do této složky vytvoř složku s*kripty* a do ní vytvoř soubor *den.txt*, do kterého zapiš aktuální datum a čas. Složku s*kripty* dvakrát zkopíruj a následně přejmenuj na *pracovni* a *pracovni2*. Vypiš, jaké složky se aktuálně nachází ve tvém adresáři. Jakmile budeš mít zkopírováno, složku *pracovni2* smaž. Změň si u svého uživatelského účtu heslo a podívej se, jaké jsou v systému spuštěné procesy.

Očekávané cíle

- Žák umí pracovat se soubory (vytvořit, smazat aj.)
- Žák umí pracovat se složkami (vytvořit, smazat aj.)
- Žák umí přecházet mezi adresáři
- Žák umí vypsat obsah adresáře
- Žák umí zobrazit obsah textového souboru
- Žák umí změnit heslo uživatele
- Žák umí vypsat spuštěné procesy

Předpoklady

- Žák umí spustit terminál

Metoda výuky

- Samostatná práce žáků

Očekávaná chybná řešení

Žák nebude schopen smazat složku, která obsahuje nějaká data. A to z důvodu, že zpočátku nezadá argument pro rekurzivní mazání dat bez dotazu o neprázdném adresáři.

8.2 Úloha 2

Téma: Úvodní problematika zápisu kódu

Název úkolu

Uvítání do programu

Obsah

V této úloze budeš muset zjistit, jak zapsat hlavičku skriptu a vyzkoušíš si textový výpis.

Zadání

Vytvoř si nový skript v Bashi a ulož ho na disk. Jakmile budeš mít vytvořený soubor, zapiš do něj skript tak, aby po jeho spuštění vypsal následující text:

„Toto je první řádek textu,

zde je druhý řádek výpisu, ale vše bylo původně zapsáno pouze v jednom řádku.“

Příkaz pro výpis zajistí vypsání obou řádků najednou. Tedy příkaz se v kódu objeví pouze jednou, nikoliv dvakrát. Stejně tak zapíšeš text do jednoho řádku bez použití enteru. Pozor také na mezeru na začátku druhého řádku.

Očekávané cíle

- Žák umí zapsat hlavičku skriptu v Bashi
- Žák umí použít příkaz echo
- Žák umí spustit skript
- Žák umí zapsat argument *„new line“* příkazu echo

Předpoklady

- Žák ovládá základní práci v Linuxu
- Žák umí spustit terminál

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žák po zápisu prvního řádku nebude vědět, jak zapsat druhý řádek bez použití opětovného použití příkazu echo a bude muset hledat, jak zapsat argument a který použít)

Očekávaná chybná řešení

U tohoto příkladu se dá očekávat, že žáci nebudou schopni pochopit, že je možné zapsat text mezi písmena, aniž by se zobrazil. Tedy argument \n většina žáků napsala až po delší době zkoušení, protože jim nešlo do hlavy, jak to zapsat, aby tam nebylo zobrazené písmeno n.

Za další chybné řešení lze brát to, že žák použije mezerník několikrát za sebou, aby se s textem dostal na další řádek. Kontrola toho řešení pak spočívá v jednoduchém roztažení okna terminálu.

Poznámky

```
# !/bin/bash

echo "Přípravka"

echo "Toto je první řádek,\ntoto je druhý řádek"
```

8.3 Úloha 3

Téma: Zápis kódu s použitím argumentů a spojením výrazů

Název úkolu

Kalendář

Obsah

V této úloze budeš muset zajistit, aby se vypsal aktuální čas a datum. Dej si však pozor na to, jak se bude datum a čas zobrazovat.

Zadání

Vytvoř skript v Bashi, který vypíše čas v českém formátu, tedy hodiny:minuty, například 12:58. Skript dále vypíše datum v tzv. krátkém tvaru, tedy například 15.03.2018.

Očekávané cíle

- Žák se naučí používat argumenty pro výpis data a času
- Žák chápe rozdílnost v zápisu velkých a malých písmen při kódování

Předpoklady

- Žák umí vytvořit skript v Bashi
- Žák umí správně použít argumenty v příkazu

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žák se dostane do problému ve chvíli, kdy zvolí špatnou variantu zápisu argumentu příkazu a dle toho se mu může stát, že výpis nedopadne dle očekávání)

Očekávaná chybná řešení

Zde lze od žáků očekávat, několik možných chyb. Zejména z důvodu velkého množství zápisu proměnných po hodiny, minuty, dny, měsíce a roky. Změna velkého písmena na malé může způsobit poměrně velkou změnu, kdy se například z měsíce stanou minuty či například rok, který by byl zapsán ve zkráceném tvaru, bude následně zapsat ve tvaru dlouhém.

Poznámky

```
# !/bin/bash
echo "Kalendář"
date +%H:%M
date +%d.%m.%Y
```

8.4 Úloha 4

Téma: Proměnné systému

Název úkolu

Není rovná se jako rovnáse

Obsah

V této úloze budeš muset zjistit, jak použít ve skriptu proměnnou, která vypíše předem zadaný obsah.

Zadání

Ve skriptu, který jsi právě obdržel, je chyba. Chceme, aby nám skript při spuštění napsal: *„Jaká barva vlasů je u holek nejhezčí? Možnosti jsou čtyři. Blond. Hnědá. Černá. Zrzavá.“* Namísto toho skript píše chybu, že příkaz nebylo možné vyhledat a nezobrazí nám odpověď.

Očekávané cíle

- Žák umí použít pevně zadanou proměnnou
- Žák chápe důležitost mezer v přiřazování hodnot proměnným

Předpoklady

- Žák umí kriticky myslet při hledání chyby
- Žák umí vyhledat význam vypsané chyby

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žáci dostanou jeden skript, ve kterém budou proměnné špatně zapsané, protože budou obsahovat mezery a budou muset zjistit, z jakého důvodu skript nezobrazí obsah proměnné, ale namísto toho vypisuje chybu, že příkaz nebylo možné vyhledat. Dále zde v kódu bude i syntaktická chyba.)

Pomůcky

- Skript s možnostmi špatného zápisu

Očekávaná chybná řešení

U tohoto příkladu lze očekávat, že žáci si budou myslet, že postačí opravit pouze jednu chybu a budou mít vyhráno. Tedy, že v předchozím příkladu zjistili, že zobrazení proměnné je možné pouze s dolarem a tady žádný dolar v zadání není. Jenže po zapsání dolaru skript stále nefunguje. Žáci tedy začnou hledat další chybu a vyzkouší mezery u znaku rovná se. Jakmile opraví chyby i zde, skript stále nezobrazí obsah proměnných. Ve chvíli, kdy žák opraví všechny chyby, skript začne teprve fungovat.

Pokud by však byl tento úkol pro některé žáky moc složitý, může učitel zvolit jednodušší variantu, kde si žák osahá, jak se s proměnnou pracuje a následně se může vrátit ke složitějšímu příkladu.

Poznámka

```
#!/bin/bash
echo "Není rovná se jako rovnáse"
Barva1=Blond
Barva2=Hnědá
Barva3=Černá
Barva4=Zrzavá

echo $barva1
echo $barva2
echo $barva3
echo $barva4
```

8.4.1 Zjednodušená varianta

U této úlohy nabízím možnost zvolit jednodušší variantu. Je zde hned několik problémů, které žák musí vyřešit a pro některé z nich to ve výuce bylo velmi komplikované, proto se zvažovala možnost změny zadání. Zadání nakonec zůstalo stejné, ale je možné zadat žákům příklad buďto tento složitější, viz kód výše, či následující kód po tímto textem. Záleží na učiteli, jakou variantu zvolí. Návrhem mohou být následující situace. První příklad dostanou všichni žáci jednodušší a následně dostanou ten složitější. Další variantou může být, že všichni dostanou ihned složitější a po nějaké době neúspěchu učitel rozhodne, komu vyměnit složitější úkol za jednodušší a po jeho vyřešení a vrátí k řešení složitějšího. Eventuálně může při zadávání příkladu učitel rovnou rozhodnout, že někteří žáci dostanou jednodušší variantu, někteří zase tu složitější.

```
#!/bin/bash
echo "Názvy procesorů"
cpu1=Intel
cpu2=Intel

echo $cpu1
echo $cpu2
```

8.5 Úloha 5

Téma: aritmetické operace

Název úkolu

Podivuhodná kalkulačka

Obsah

V této úloze budeš muset zjistit, proč kalkulačka počítá nějak divně. Nefunguje tam totiž ani jedna operace tak, jak by správně měla.

Zadání

Ve skriptu *kalkulacka.sh* zjisti, proč ani jedna operace neudělá dle komentáře to, co by dělat měla. Nezapomeň na začátku upravit syntaxi zápisu pro výpočet aritmetických operací. Zjisti, kde je v prvních úlohách problém, že ani jedna operace nefunguje a jakmile to zjistíš, zaměř se na předposlední příklad, na dělení. Vyjde výsledek při tomto zápisu přesně tak, jak očekáváš?

Očekávané cíle

- Žák správně používá ve skriptech aritmetické operace
- Žák rozezná zápis obyčejného textu a příkazu

Předpoklady

- Žák zná základní aritmetické počty
- Žák umí používat proměnné

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žáci obdrží skript, ve kterém budou mít pět příkladu, ale ani jeden nebude fungovat, tedy budou muset zjistit, jaký zápis pro výpočet je správný, opravit ho a následně i objevit, jak zapsat znaménka, aby vyšel výsledek dle zadání)

Pomůcky

- Skript s příklady

Očekávaná chybná řešení

Opět se setkáváme s problémem, kdy žáci musí naleznou chybu v zápisu proměnných při aritmetickém počítání. Očekává se, že žáci v tomto příkladu budou neustále chybně zkoušet zapisovat znak proměnné, dokud se jim nepodaří odstranit alespoň jednu chybu a zobrazit správný výsledek. Další očekávanou chybou je, že žáci použijí aritmetická znamínka ve špatné formě.

Poznámka

```
# !/bin/bash
echo "Podivuhodná kalkulačka"

cislo1=10
cislo2=2

# sečti cislo1 a cislo2 (výsledek: 12)
echo "$cislo1 plus $cislo2 = $(($cislo1+$cislo2))"

# Od proměnné cislo1 odečti cislo2 (výsledek: 8)
echo "$cislo1 minus $cislo2 = $(($cislo1-$cislo2))"

# Proměnnou cislo1 vynásob s proměnnou cislo2 (výsledek 20)
echo "$cislo1 krát $cislo2 = $(($cislo1*$cislo2))"

# Proměnnou cislo1 vyděl proměnnou cislo2 (výsledek: 5)
echo "$cislo1 děleno $cislo2 = $(($cislo1/$cislo2))"

# Výsledek je?
echo "$vysledek = $(($cislo1+$cislo2/$cislo2))"
```

8.6 Úloha 6

Téma: Podmínky

Název úkolu

Je dopoledne nebo odpoledne?

Obsah

V této úloze budeš muset naprogramovat skript tak, aby po spuštění vypsal, v jaké části dne se zrovna nacházíme.

Zadání

Vytvoř si nový skript *castdne.sh* a vymysli ho tak, aby po spuštění vypsal, zda je před polednem či po poledni.

Očekávané cíle

- Žák umí použít podmínku
- Žák chápe důležitost mezer v zápisu podmínky

Předpoklady

- Žák rozumí pojmu proměnná
- Žák umí proměnnou použít
- Žák rozumí pojmu podmínka
- Žák rozumí pojmu podmíněný příkaz

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žák nezná, jak zapsat podmínku a její možnosti)

Očekávaná chybná řešení

Očekává se, že žák udělá chybu ve znaménku větší / menší.

Poznámka

- Žák musí zjistit, jak zapsat do lokální proměnné systémový čas
- Žák musí zjistit, jak porovnat podmínkou aktuální čas a dle toho vypsat, o jakou část dne se jedná (vytvořit podmínku, zda je)
- Na základě tohoto prvního příkladu pak žáci budou schopni vytvořit i složitější podmínky pro více částí dne.

```
#!/bin/bash
echo "Je dopoledne nebo odpoledne?"

H=$(date +%H)

if [ $h -lt 12 ]; then
 echo "Je před polednem"

else
 echo "Je po poledni"
fi
```

8.7 Úloha 7

Téma: Podmínky

Název úkolu

Je dopoledne, odpoledne nebo poledne?

Obsah

V této úloze budeš muset naprogramovat skript tak, aby po spuštění vypsal, jestli se nacházíme před polednem, po poledni nebo je právě poledne.

Zadání

Vytvoř si nový skript *castdne2.sh* a vymysli ho tak, aby po spuštění vypsal, zda je před polednem, po poledni či je právě poledne. Nezapomeň, že poledne je jen určitou chvíli.

Očekávané cíle

- Žák umí použít podmínku
- Žák chápe důležitost mezer v zápisu podmínky
- Žák umí logicky posoudit, jak podmínku vytvořit, aby určil časové rozmezí

Předpoklady

- Žák rozumí pojmu proměnná
- Žák umí použít proměnnou
- Žák rozumí pojmu podmínka

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žák neví, jaké argumentu příkazu je možné nastavit, aby podmínka byla rozvětvená, eventuálně byla složená z více částí)

Očekávaná chybná řešení

Žák musí vymyslet, jak porovnat nejen hodiny, ale i minuty. Před polednem a po poledni může zůstat stále pouze u porovnávání hodin, nicméně u pravého poledne je nutné porovnat i minuty. A to je problém, se kterým se bude žák muset popasovat, protože musí zjistit, jak hodiny zapsat, aby je bylo možné porovnat.

Poznámka

- Žák musí zjistit, jak porovnat podmínkou aktuální čas a dle toho vypsat, o jakou část dne se jedná (vytvořit podmínku, zda je před polednem, po poledni nebo je právě poledne)
- Žák následně konfliktem zjistí, že není možné porovnávat pouze hodiny, ale je nutné porovnat i minuty, protože by bylo poledne déle než pouze ve 12:00

```
#!/bin/bash
echo "Je dopoledne, odpoledne nebo poledne?"

H=$(date +%H)
M=$(date +%M)

echo $h$m

if [ $h -lt 12 ]; then
 echo "Je před polednem"

 elif [ $h$m -eq 1200 ]; then
echo "Je poledne"

else
 echo "Je po poledni"
fi
```

8.8 Úloha 8

Téma: Podmínky

Název úkolu

Jaká je část dne?

Obsah

V této úloze budeš muset naprogramovat skript tak, aby po spuštění vypsal, v jaké části dne se zrovna nacházíme.

Zadání

Vytvoř si nový skript *castdne.sh* a vymysli ho tak, aby po spuštění vypsal, zda je noc, dopoledne, poledne, odpoledne, večer či půlnoc. Nezapomeň, že poledne a půlnoc je jen určitou chvíli. Dopoledne, odpoledne a večer si rozvrhni dle znalostí, v jakých časech se tyto části dne nachází.

Očekávané cíle

- Žák umí použít podmínku
- Žák chápe důležitost mezer v zápisu podmínky
- Žák umí logicky posoudit, jak podmínku vytvořit, aby určil časové rozmezí

Předpoklady

- Žák rozumí pojmu proměnná
- Žák umí použít proměnnou
- Žák rozumí pojmu podmínka

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žák neví, jaké argumentu příkazu je možné nastavit, aby podmínka byla rozvětvená, eventuálně byla složená z více částí)

Očekávaná chybná řešení

V tomto příkladu se musí žák zaměřit na kombinaci podmínek. Tedy když žák zadal operand „or“ namísto „and“, výsledek se mu nezobrazil správně, i když měl celý skript správně.

Poznámka

- Dle složitosti příkladu předpokládám, že i velmi talentovaní žáci tento komplikovaný a vrstvený problém mohou řešit i několik hodin.

```
#!/bin/bash
echo "Jaká je část dne?"

H=$(date +%H)
M=$(date +%M)

echo "$h:$m"

if [ $h$m -eq 0000 ]; then
 echo "Je půlnoc"
 elif [ $h$m -eq 1200 ]; then
 echo "Je poledne"
 elif [ $h$m -gt 0000 -a $h$m -lt 0800 ]; then
 echo "Je noc"
 elif [ $h$m -gt 0800 -a $h$m -lt 1200 ]; then
 echo "Je dopoledne"
 elif [ $h$m -gt 1200 -a $h$m -lt 2000 ]; then
 echo "Je odpoledne"
 elif [ $h$m -gt 2000 -a $h$m -lt 2359 ]; then
 echo "Je noc"
else
 echo CHYBA
fi
```

8.9 Úloha 9

Téma: Výběr dat

Název úkolu

Výběr skladeb oblíbeného interpreta

Obsah

V této úloze budeš muset naprogramovat skript tak, aby bylo možné zadat jméno oblíbeného interpreta a skript vypsal do terminálu názvy jeho skladeb.

Zadání

Vytvoř si nový skript i*nterpret.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a následně vypsal veškeré jeho skladby ze seznamu do okna terminálu. Zkontroluj, zda jsi vybral opravdu všechny.

Očekávané cíle

- Žák umí použít příkaz grep
- Žák umí vybrat správný argument příkazu grep
- Žák chápe důležitost velkých a malých písmen v zápisu výběru dat
- Žák umí použít přesměrování výstupu
- Žák umí zadat vstup z klávesnice do proměnné

Předpoklady

- Žák rozumí pojmu echo
- Žák umí použít echo

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žák neví, jak poskládat příkaz pro očekávané zobrazení)

Očekávaná chybná řešení

Chybové řešení v tomto příkladu se očekává ve chvíli, kdy žák musí správně poskládat příkaz grep. Pokud nepoužije uvozovky při vyhledávání interpreta, nedostane přesné výsledky. Jedná se zejména o dvou a víceslovná jména. U jednoslovných jmen interpretů by rozdíl poznat nebyl. Tato chyba přetrvává u všech následujících skriptů s podobnou tématikou.

Poznámka

- Příprava pro následující příklad s počítáním.
- Větší problém vznikne ve chvíli, kdy žák porovná zdrojový soubor s výsledkem, protože některé řádky nezačínají velkým písmenem, proto bude muset naleznout, z jakého důvodu a doplnit argument pro ignorování velkosti písmen
- Dávat proměnnou do uvozovek je poměrně nezvyklé a v této sbírce ojedinělé. Důvodem k tomu je ochrana při vyhledávání víceslovného názvu. Pokud by se v seznamu vyhledávaly pouze jednoslovné názvy, pak uvozovky být nemusí.

```
#!/bin/bash
echo "Výběr skladeb oblíbeného interpreta"
echo "Zadej jméno interpreta"
read "interpret"
grep "$interpret" interpreti.txt
```

8.10 Úloha 10

Název úkolu

Výběr skladeb oblíbeného interpreta s pojistkou

Obsah

V této úloze budeš muset naprogramovat skript tak, aby bylo možné zadat jméno oblíbeného interpreta a skript vypsal názvy jeho skladeb.

Zadání

Vytvoř si nový skript *interpretp.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a následně vypsal veškeré jeho skladby ze seznamu. Zkontroluj, zda jsi vybral opravdu všechny.

Očekávaná chybná řešení

U tohoto příkladu žáci mohou chybovat v tom, že použijí špatný argument a skript nezobrazí veškeré skladby od interpreta, protože některé řádky nemusí začínat velkým písmenem.

Poznámka

- Žák umí použít argument -i
- Rozšiřující úloha: některé řádky nemusí začínat velkým písmenem, proto bude muset žák zjistit, z jakého důvodu se nemusely vybrat všechny skladby a doplnit argument pro ignorování velkosti písmen

```
#!/bin/bash
echo "Výběr skladeb oblíbeného interpreta s pojistkou"
echo "Zadej jméno interpreta"
read "interpret"
grep "$interpret" -I interpreti.txt
```

8.11 Úloha 11

Téma: výběr dat do souboru

Název úkolu

Skladby oblíbeného interpreta do souboru

Obsah

V této úloze budeš muset naprogramovat skript tak, aby bylo možné zadat jméno oblíbeného interpreta a skript zapsal veškeré skladby do nově vytvořeného textového souboru.

Zadání

Vytvoř si nový skript *interpretzapis.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a následně zapsal všechny skladby do nového textového souboru *skladby.txt*.

Očekávané cíle

- Žák vytvoří nový soubor
- Žák zapíše vybraná data do textového souboru

Předpoklady

- Žák rozumí příkazu grep
- Žák umí použít příkaz grep

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žák neví, jak zapsat data do textového souboru)

Očekávaná chybná řešení

Žák bude mít problém se zápisem do nového souboru. Nejprve si soubor ručně vytvoří a až následně do něj bude zapisovat, protože neví, že je možné soubor vytvořit přímo ve skriptu.

Poznámka

```
#!/bin/bash
echo "Skladby oblíbeného interpreta do souboru"
echo "Zadej jméno interpreta"
read "interpret"
grep "$interpret" -I interpreti.txt > skladby.txt
```

8.12 Úloha 12

Téma: Výběr dat a počet výskytů

Název úkolu

Počet skladeb oblíbeného interpreta

Obsah

V této úloze budeš muset naprogramovat skript tak, aby bylo možné zadat jméno oblíbeného interpreta a skript zapsal skladby do textového souboru a zároveň spočítal počet jeho skladeb.

Zadání

Vytvoř si nový skript *interpretpocet.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a následně zapsal počet skladeb ze seznamu do textového souboru ze seznamu.

Očekávané cíle

- Žák umí použít příkaz wc
- Žák umí použít rouru v Linuxu

Předpoklady

- Žák rozumí příkazu grep
- Žák umí použít příkaz grep

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žák neví, jak automaticky spočítat počet výskytů vyhledávaného řetězce)

Očekávaná chybná řešení

Žák se dostane do problému ve chvíli, kdy bude zadávat příkaz pro spočítání řádků. Musí použít *rouru*, jinak se mu počet řádků nezobrazí v jednoduchém tvaru díky příkazu *wc*. Nelze ale považovat za chybné řešení, kdy žáci nepoužijí příkaz *wc*, ale pouze argument *–c* k příkazu grep.

Poznámka

- Žák může počet řádků spočítat i pomocí argumentu –c (count) příkazu grep

```
#!/bin/bash
echo "Počet skladeb oblíbeného interpreta"
echo "Zadej jméno interpreta"
read "interpret"
grep "$interpret" -I interpreti.txt | wc > skladby.txt
```

8.13 Úloha 13

Téma: Výběr dat a počet výsledků

Název úkolu

Počet skladeb oblíbeného interpreta

Obsah

V této úloze budeš muset naprogramovat skript tak, aby bylo možné zadat jméno oblíbeného interpreta a skript vypsal počet skladeb do okna terminálu.

Zadání

Vytvoř si nový skript *interpretpocetT.sh*, který spočítá počet skladeb. Vymysli ho tak, aby se po spuštění zeptal na jméno interpreta, uložil výsledek do textového souboru a následně obsah souboru vypsal do okna terminálu. Pro spočítání ale nepoužívej k vyhledávacímu příkazu žádný argument, ale kombinaci několika příkazů.

Očekávané cíle

- Žák umí použít příkaz wc
- Žák umí použít rouru v Linuxu

Předpoklady

- Žák rozumí příkazu grep
- Žák umí použít příkaz grep
- Žák umí použít příkaz cat

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žák neví, jak automaticky spočítat počet výskytů vyhledávaného řetězce a následně spojit s příkazem pro zobrazení obsahu souboru)

Očekávaná chybná řešení

Zde se žáci dostanou do problému ve chvíli, kdy budou muset zobrazit obsah souboru, do které právě uložili získaná data. Sice výsledky uloží, ale již je nezobrazí. Zejména z toho důvodu, že je nenapadne možná kombinace příkazů použitím středníku.

Poznámka

```
#!/bin/bash
echo "Počet skladeb oblíbeného interpreta"
echo "Zadej jméno interpreta"
read "interpret"
grep "$interpret" -I interpreti.txt | wc > skladby.txt;
cat skladby.txt
```

8.14 Úloha 14

Téma: Výběr dat

Název úkolu

Výběr skladeb druhého oblíbeného interpreta

Obsah

V této úloze budeš muset naprogramovat skript tak, aby bylo možné zadat jméno druhého oblíbeného interpreta a skript připsal na konec textového souboru názvy skladeb.

Zadání

Vytvoř si nový skript *druhyinterpret.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a následně připsal veškeré jeho skladby ze seznamu na konec textového souboru s*kladby.txt*. Soubor by měl po zápisu obsahovat skladby obou interpretů.

Očekávané cíle

- Žák umí použít příkaz grep
- Žák umí vybrat správný argument příkazu grep
- Žák chápe důležitost velkých a malých písmen v zápisu výběru dat
- Žák dokáže použít přesměrování výstupu

Předpoklady

- Žák rozumí pojmu echo
- Žák umí použít echo

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žák neví, jak poskládat příkaz pro očekávaný zápis)

Očekávaná chybná řešení

U tohoto příkladu žák s největší pravděpodobností přepíše původní obsah. Stane se tak zejména z důvodu, že použije pouze jeden znak špičaté závorky, tedy pro zápis do souboru.

Poznámka

```
#!/bin/bash
echo "Výběr skladeb druhého oblíbeného interpreta"
echo "Zadej jméno druhého interpreta"
read "interpret"
grep "$interpret" -I interpreti.txt >> skladby.txt
```

8.15 Úloha 15

Téma: výběr dat a přesměrování výstupu

Název úkolu

Skladby oblíbeného interpreta do textového dokumentu i do terminálu

Obsah

V této úloze budeš muset naprogramovat skript tak, aby bylo možné zadat jméno oblíbeného interpreta a skript zapsal všechny skladby do nového textového dokumentu a zároveň je vypsal do terminálu.

Zadání

Vytvoř si nový skript *interpretTerminal.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a vytvořil textový dokument, do kterého zapíše názvy všech jeho skladeb a zároveň se nalezené skladby zobrazí na obrazovce terminálu.

Očekávané cíle

- Žák umí použít & pro kombinaci příkazů

Předpoklady

- Žák rozumí příkazu grep
- Žák umí použít příkaz grep

Metoda výuky

- Samostatná práce žáků

Poznámka

- Rozšíření předchozích úloh
- Žáci si vyzkouší, jak je možné zobrazit výpis do terminálu a zároveň ho zapsat do textového souboru

```
#!/bin/bash
echo "Skladby oblíbeného interpreta do textového dokumentu i do terminálu"
echo "Zadej jméno interpreta"
read "interpret"
grep "$interpret" -I interpreti.txt > skladby.txt; cat skladby.txt
```

8.16 Úloha 16

Téma: Výběr dat a přesměrování výstupu

Název úkolu

Skladby oblíbeného interpreta do textového dokumentu i do terminálu

Obsah

V této úloze budeš muset naprogramovat skript tak, aby bylo možné zadat jméno oblíbeného interpreta a skript zapsal všechny skladby do nového textového dokumentu a zároveň je vypsal do terminálu.

Zadání

Vytvoř si nový skript *interpretTerminal2.sh* a vymysli ho tak, aby se po spuštění zeptal na jméno interpreta a vytvořil textový dokument, do kterého zapíše názvy všech jeho skladeb a zároveň se nalezené skladby zobrazí na obrazovce terminálu. Nepoužívej však kombinaci příkazů jako v předešlé úloze.

Očekávané cíle

- Žák umí použít příkaz tee

Předpoklady

- Žák rozumí příkazu grep
- Žák umí použít příkaz grep
- Žák umí použít rouru

Metoda výuky

- Samostatná práce žáků

Očekávaná chybná řešení

U tohoto příkladu se u žáků předpokládá, že se pokusí zadat název souboru, do které se mají data zapsat a následně název souboru, ze kterého se mají data přečíst. Tak tomu bylo v předchozích příkladech. Nejprve bylo nutné data načíst, uložit a následně zobrazit. Příkaz *tee* udělá uložení a zobrazení najednou.

Poznámka

- Rozšíření předchozích úloh pro zopakování
- Obdobná úloha pro použití roury, pokud by žáci při počítání skladeb obešli příkaz wc argumentem –c příkazu grep

```
#!/bin/bash
echo "Skladby oblíbeného interpreta do textového doku-
mentu i do terminálu"
echo "Zadej jméno interpreta"
read "interpret"
grep -i "$interpret" interpreti.txt | tee skladby.txt
```

8.17 Úloha 17

Téma: Vyhledání a filtrování souborů

Název úkolu

Kdo hledá, najde

Obsah

V této úloze budeš muset naprogramovat skript, díky kterému ve směsi textových souborů najdeš pouze ty smysluplné.

Zadání

Vytvoř si nový skript *hledejsmysl.sh* a vymysli ho tak, aby ze všech souborů ze složky s*mes* vybral pouze ty soubory, které obsahují v názvu souboru text *vtip*. Jakmile je najdeš, zobraz si jejich obsah.

Očekávané cíle

- Žák rozumí příkazu find
- Žák umí použít příkaz find

Předpoklady

- Žák rozumí příkazu grep
- Žák umí použít příkaz grep
- Žák umí použít rouru

Metoda výuky

- Samostatná práce žáků
- Pedagogický konflikt (žák nezná příkaz pro vyhledávání v řetězcích)

Očekávaná chybná řešení

Zde je velmi pravděpodobně, že žák zadá špatně název vyhledávaného souboru. Buďto nezadá hvězdičky, ale jiné znaky nebo nevyplní hvězdičky z obou stran u slova *vtip*. Kvůli tomu pak nezíská všechny soubory, které klíčové slovo obsahují. Dále také mohou žáci chybovat v zadávání místa, ve kterém bude skript prohledávat soubory s daným názvem.

Poznámka

```
#!/bin/bash
echo "Kdo hledá, najde"
find . -name "*vtip*"
```

9 Závěr

Tato diplomová práce byla vytvořena za účelem poskytnout sbírku úloh zaměřenou na problémovou výuku základní práce v Bashi. Důvod k tomuto rozhodnutí byl zejména po vyhledávání příkladů pro přípravu výuky na středních školách, na kterých jsem měl vyučovat základní práci v Bashi. Po určitém čase vyhledávání vyhovujících publikací jsem dospěl k názoru, že existují sice knihy zaměřené na problémovou výuku nebo knihy, které řeší skriptování v Bashi, ale jen velmi zřídka jsem nacházel příklady, které by byly zadané problémově tak, aby byly pro žáky motivující a měly v řešení nějaký smysl a návaznost. Proto jsem se při tvorbě vlastních výukových příkladů rozhodl, že zpracuji svou sbírku problémových úloh zaměřených na toto téma, abych tuto oblast alespoň trochu obohatil o zajímavé příklady.

Problémové úlohy mohou přinést různé zajímavé poznatky, zejména ve chvíli, kdy jsou úlohy zadány zajímavě, protože pak má žák velkou potřebu objevovat a je motivován k další práci. Největším plusem na této metodě je asi ale to, že se žáci vlastním objevováním naučí mnohem více, než pouhým bezhlavým opisováním.

V praktické části jsem se zaměřil zejména na tvorbu úloh takových, aby měly smysl a žáci pouze bezhlavě neplnili zadání. Je tomu tak zejména z důvodu motivace žáků k vyřešení problémových úloh. Každý úkol z této sbírky, kterých je celkem 17, byl otestován nadvakrát ve výuce, protože první verzi vyzkoušela první skupina žáků, na které se ověřilo, zda zadání plní účel. Pokud tomu tak nebylo, zadání příkladu bylo přepracováno a otestováno ve druhé skupině žáků. Následně veškeré postřehy byly zaznamenány jako poznámky pro učitele, který by chtěl danou úlohu ve své výuce použít. Nejčastějším problém u všech úloh byl časový rozdíl v řešení příkladů žáky. Většinou jsem očekával rozdíl v minutách a reálně byl rozdíl spíš v desítkách minut. Nicméně asi žádná z negativních skutečností mi nezkazila radost ve chvíli, kdy se žák nad příkladem rozčiloval a po několika minutách rozplýval. Byly chvíle, kdy žáci řekli, že už to dělat nebudou, že to vyřešit prostě nelze a po několika minutách někdo vykřikl, že se mu to povedlo a všichni ostatní s údivem začali tisknout jednu klávesu za druhou, jen aby se jim to také podařilo vyřešit.

Dovolím si tedy tvrdit, že ve chvíli, kdy jsem učil o rok dříve stejnou látku frontálně a žáci si pouze z tabule opisovali a nemuseli přemýšlet nad kódem, tak poté při písemném zkoušení byly obrovské rozdíly. Žáci, kteří prošli problémovou výukou měli o

mnoho lepší výsledky. Zejména bych se přikláněl i k tomu, že věděli, jak zadat klíčová slova do vyhledávače, který jim pomohl s vyřešením problému.

Návaznost na tuto práci vidím zejména ve dvou možnostech. První z nich je kvantitativní ověření sbírky úloh ve školách, tedy ověření na větším vzorku žáků než pouze na dvou skupinách. I když jsem měl možnost udělat první testovací verzi a následně druhou verzi pro druhou skupinu a porovnat výsledky, stále si myslím, že by se dalo u některých úloh naleznout i nějaké jiné řešení, změnit zadání či například poupravit skript pro žáky. Z tohoto důvodu by bylo kvantitativní ověření sbírky úloh velmi zajímavou prací. Druhou možností by mohlo být vytvoření složitějších úloh, které by se zabývaly již pokročilejšími příklady v Bashi.

Literatura

[1] *Rámcový vzdělávací program pro základní vzdělávání*. [online]. Praha: MŠMT, 2016 [cit. 2018-02-25]. Dostupné z: http://www.nuv.cz/uploads/RVP_ZV_2016.pdf

[2] KOPKA, Jan, 2005. V*ýzkumné strategie při řešení problémů*. Ústí nad Labem: univerzita J. E. Purkyně. Dostupné také z: pdf.truni.sk/download?Zbornik/smolenice/kopka.pdf

[3] PRŮCHA, Jan, eliška WALTEROVÁ a Jiří MAREŠ, 2001. *Pedagogický slovník*. 3., rozš. A aktualiz. Vyd. Praha: Portál. Isbn 80-717-8579-2.

[4] MAŇÁK, Josef a vlastimil ŠVEC, 2003. V*ýukové metody*. Brno: Paido. Isbn 80-731-5039-5.

[5] SKALKOVÁ, Jarmila, 2007. *Obecná didaktika: vyučovací proces, učivo a jeho výběr, metody, organizační formy vyučování*. 2., rozš. A aktualiz. Vyd., [V nakl. Grada] vyd. 1. Praha: Grada. Pedagogika (Grada). Isbn 978-80-247-1821-7.

[6] KLIČKOVÁ, Marie. *Problémové vyučování ve školní praxi*. Praha: státní pedagogické nakladatelství, 1989. Isbn 80-04-23522-0.

[7] DUŠEK, Pavel. *Tvorba výukových materiálů pro výuku Linuxu*. České Budějovice: Jihočeská univerzita v Českých Budějovicích, 2009. Bakalářská práce.

[8] VALADE, Janet. *Linux: jdi do toho*. Praha: Grada, 2006. Průvodce (Grada). Isbn 80-247-1455-8.

[9] FLICKENGER, Rob. *Linux server na maximum: 100 tipů a řešení pro náročné*. Brno: CP Books, 2005. Isbn 80-251-0586-5.

[10] SHAH, steve a Wale soyinka. A*dministrace systému Linux: překlad čtvrtého vydání*. Praha: Grada, 2007. Isbn 978-80-247-1694-7.

[11] SIEVER, ellen. *Linux v kostce: pohotová referenční příručka*. Praha: Computer Press, 1999. Isbn 80-7226-227-0.

[12] *SS64* [online]. [cit. 2018-04-14]. Dostupné z: https://ss64.com/bash/date.html

[13] PROCHÁZKA, Roman. *Teorie a praxe poradenské psychologie*. Praha: Grada, 2014. Psyché (Grada). Isbn 978-80-247-4451-3.

[14] BURTCH, ken O. *Linux shell scripting with Bash*. Indianapolis, ind.: sams, 2004. Isbn 0672326426.

[15] *Linux: dokumentační projekt*. 4., aktualiz. Vyd. Přeložil Lubomír PTÁČEK. Brno: Computer Press, 2007. Isbn 978-80-251-1525-1.

[16] MASTERS, Jon a Richard BLUM. *Linux profesionálně: programování aplikací*. Brno: zoner Press, 2008. Encyklopedie zoner Press. Isbn 978-80-86815-71-8.

[17] MATTHEW, Neil a Richard stones. *Linux: začínáme programovat*. Brno: Computer Press, 2008. Isbn 978-80-251-1933-4.

Seznam použitých zkratek

GNOME = GNU Network Object Model environment

GUI = Grafické uživatelské rozhraní

KDE = Common Desktop environment

RVP = Rámcový vzdělávací plán

RVP ZV = Rámcový vzdělávací plán pro základní vzdělávání

Seznam obrázků a tabulek

Přílohy

Veškeré skripty a texty je možné naleznout také na přiloženém DVD k této práci.

9.1 Skripty pro učitele

Pod následujícím odkazem je možné stáhnout fungující vypracované skripty pro učitele.

Http://zakladyvbashi.8u.cz/ucitel.zip

9.2 Skripty pro žáky

Pod následujícím odkazem je možné stáhnout připravené skripty pro žáky.

Http://zakladyvbashi.8u.cz/zak.zip

9.3 Soubory k vyhledávání

Pod následujícím odkazem je možné stáhnout složku s*mes* se soubory, které jsou k poslední úloze zaměřené na vyhledávání souborů.

Http://zakladyvbashi.8u.cz/smes.zip

9.4 Soubor s interprety pro vyhledávání

Pod následujícím odkazem je možné stáhnout soubor, který obsahuje data pro vyhledávání textu v souborech.

Http://zakladyvbashi.8u.cz/interpreti.zip

9.5 Zadání všech příkladů

Pod následujícím odkazem je možné stáhnout soubor, který obsahuje zadání všech příkladů.

Http://zakladyvbashi.8u.cz/zadani.zip