

Sem vložte zadání Vaší práce.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
KATEDRA TEORETICKÉ INFORMATIKY



Diplomová práce

Comma-shell, interaktivní debugger shellu

Bc. Tomáš Nesrovnal

Vedoucí práce: Ing. Jan Baier

28. března 2017

Poděkování

Doplňte, máte-li komu a za co děkovat. V opačném případě úplně odstráňte tento příkaz.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 28. března 2017

.....

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2017 Tomáš Nesrovnal. Všechna práva vyhrazena.

Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Nesrovnal, Tomáš. *Comma-shell, interaktivní debugger shellu*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017. Dostupný také z WWW: (<https://github.com/nesro/nesrotom-dip-2016>).

Abstrakt

V několika větách shrňte obsah a přínos této práce v češtině. Po přečtení abstraktu by se čtenář měl mít čtenář dost informací pro rozhodnutí, zda chce Vaši práci číst.

Klíčová slova Nahradte seznamem klíčových slov v češtině oddělených čárkou.

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords Nahradte seznamem klíčových slov v angličtině oddělených čárkou.

Obsah

Úvod	1
1 Definice a pojmy	3
2 Historie shellu a dnešní využití	5
2.1 Bash - Bourne Again Shell	5
2.2 Sh - Bourne Shell	5
2.3 Dash - Debian Almquist Shell	5
2.4 Zsh - Z Shell	5
2.5 Práce v příkazové řádce	5
2.6 Nastavení shellu	6
2.7 Terminál	6
3 Cíl práce	7
3.1 Motivace	7
3.2 Existující řešení	7
3.3 Bash frameworky pro psaní skriptů	7
3.4 Bash frameworky pro správu doplňků	8
3.5 Bash frameworky pro úpravu promptu	8
4 Analýza a návrh	9
4.1 Historie UNIXu	9
4.2 Historie shellu	9
4.3 Fungování shellu	9
4.4 Debugování shellu	10
4.5 Možnosti debugování v interaktivním shellu	11
4.6 Logování výstupu	12
4.7 Automatizované spouštění příkazů	12
4.8 Chování uživatelů v příkazové řádce	12
4.9 Statická analýza skriptů	12

5	Realizace	15
5.1	Nespouštění příkazů	15
5.2	Implementace debuggeru	15
5.3	Hooks	15
5.4	Bezpečný mód	16
5.5	Historie	17
	Závěr	19
	Literatura	21
A	Seznam použitých zkratk	23
B	Obsah příloženého CD	25

Seznam obrázků

4.1	Historie Shellu	9
-----	---------------------------	---

Úvod

Grafické uživatelské rozhraní (GUI) se jednoduše ovládá, ale ne vždy je k dispozici. To platí zejména při ovládání serverů.

Rozhraní příkazové řádky (CLI) je základní textové prostředí pro komunikaci s operačním systémem. Umožňuje spouštění programů, vkládat vstupní data a sledovat výstupní data v terminálu.

Jedním ze základních bodů UNIXové filosofie je mít jednoduché programy, které dělají pouze jednu věc, ale dělají ji dobře. To platí zejména pro základní příkazy ze sady GNU coreutils, tedy příkazy pro základní manipulaci se soubory, shellem a textem.

Tyto základní příkazy je možné řetězit a tím vytvářet užitečné jednořádkové skripty.

TODO: napsat o tom, že pro začátečníka to může být neintuitivní, musí si pamatovat spoustu příkazů. o návratových kódech, o historii příkazů a o logování

Definice a pojmy

prompt shell interaktivni debugger gnu linux root hook terminal

todo: spousta veci nema cesky ekvivalent, zde bych mohl tyto pojmy vysvetlit.

Historie shellu a dnešní využití

Napsat o tom, že bash je dneska nejcastější, protože linux je vlastně GNU/Linux a bash je GNU bash. Zsh je lepší snad úplně ve všem. Sh, resp Dash je rychlejší.

2.1 Bash - Bourne Again Shell

https://cs.wikipedia.org/wiki/Debian_Almquist_shell <https://wiki.archlinux.org/index.php/Dash>

2.2 Sh - Bourne Shell

todo

2.3 Dash - Debian Almquist Shell

todo

2.4 Zsh - Z Shell

todo

<http://zsh.sourceforge.net/FAQ/>

2.5 Práce v příkazové řádce

todo: tady bych chtěl shrnout obecně to, že se píšou příkazy jako text a uživatel vše vidí jako text. co z toho plyne za následky a jak je snadné udělat chybu

2.6 Nastavení shellu

GNU/Linuxové distribuce, které nabízejí předpřipravené prostředí, mají pro výchozí shell bash připravené takzvané rc soubory, které nějakým způsobem upravují běžící interaktivní shell. Jednou z nejviditelnějších změn je nastavení proměnné PS1, o které bude řeč v pozdější kapitole, která určuje styl promptu. Ve výchozím nastavení bashe, tedy po spuštění bashe bez načtení rc souborů (`bash -norc`), se v PS1 zobrazuje pouze název shellu, jeho verze a zdali je uživatel root. Po načtení výchozích rc souborů se v PS1 ukazují informace jako je například jméno uživatele, jméno počítače a co je velmi důležité, jméno aktuálního adresáře.

Tyto informace jdou snadno získat použitím některých základních příkazů, například `whoami`, `hostname`, `pwd`. Protože tyto informace jsou důležité a potřebujeme je vědět pořád, chceme je mít pořád na očích.

2.7 Terminál

todo: asi by se hodilo napsat i něco o tech programech, ve kterých shell beží

Cíl práce

Napsat o tom, že cílem práce je usnadnit práci v příkazové řádce a sepsat základní funkcionalitu debuggeru.

3.1 Motivace

todo: ?

3.2 Existující řešení

Mezi seznamem různých zajímavých řešeních `awesome` (github.com/sindresorhus/awesome) existuje i podsektce `awesome-shell` (github.com/alebcay/awesome-shell), ve které lze nalézt spoustu užitečných nástrojů pro práci s příkazovou řádkou, nebo psaním skriptů v `bash`i.

Spousta těchto nástrojů je nad rámec této práce.

3.3 Bash frameworky pro psaní skriptů

Existuje mnoho projektů, jejichž cílem je vytvořit framework v `bash`i, který má nějakým způsobem zjednodušit vytváření, především větších, skriptů v `bash`i. Spousta dnešních vývojářů je zvyklá na objektově orientovaný přístup k programování a je tedy pro těžké vytvořit větší program, který by zůstal přehledný.

3.3.1 Bash OO framework

`Bash OO framework` (github.com/niieani/bash-oo-framework) je framework napasný v `bash`i, který umožňuje vytvářet třídy, výjimky a testy. Jeho cílem je vytvořit prostředí pro psaní skriptu, kde se bude snadněji psát čitelný kód, bez částí, které se opakují.

3.4 Bash frameworky pro správu doplňků

Existuje spousta věcí, které uživatel příkazové řádky potřebuje občas řešit. Řešení pro daný problém je víc. Buď si uživatel vyřeší problém sám, nebo bude hledat řešení na internetu. Shellové frameworky, které umějí i spravovat doplňky, mohou mít řešení pro daný problém. Instalace a použití pak bude velmi jednoduché, intuitivní a bude zde nějaká záruka o funkčnosti a kvalitě.

Přestože je tato práce převážně o shellu bash, zsh je v tomto mnohem rozšířenější.

3.4.1 Oh My Zsh

Oh My Zsh ([ohmyz.sh](https://github.com/ohmyzsh/ohmyzsh)) je framework pro Zsh.

3.4.2 Bash-it

Bash-it (<http://github.com/Bash-it/bash-it>)
(itsfoss.com/bash-it-terminal-tool/)

3.5 Bash frameworky pro úpravu promptu

3.5.1 Liquid Prompt

Projekt liquidprompt github.com/nojhan/liquidprompt je adaptivní prompt v interaktivním bashi.

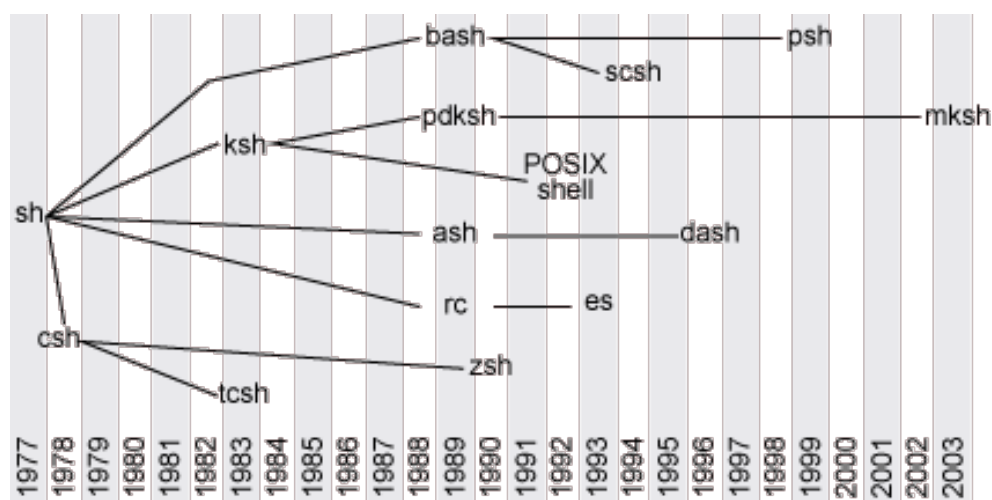
3.5.2 Sexy Bash Prompt

Projekt sexy-bash-prompt <https://github.com/twolfson/sexy-bash-prompt> je další používaný prompt v interaktivním bashi.

Analýza a návrh

4.1 Historie UNIXu

4.2 Historie shellu



Obrázek 4.1: Historie Shellu

V latexových komentářích jsou nějaké odkazy o historii.

4.3 Fungování shellu

4.3.1 Životní cyklus příkazu

todo: popsat, co všechno se stane, od napsání příkazu až po jeho vykonání

4.3.2 Gramatika shellu

Soubor s gramatikou bashe, `parse.y`, má přes 6000 řádek. Chtěl bych zde napsat zjednodušenou gramatiku, která by se dala snadno pochopit (ono to zas tak složité není).

Popsat jakým způsobem parsuje gramatiku BASH (`yacc`) a jakým to dělají parsery `bashlex` a `bashast`.

4.3.2.1 Projekty parsující shell

<https://github.com/bemeurer/beautysh/blob/master/beautysh/beautysh.py>

4.3.2.2 Bashlex

<https://github.com/idank/bashlex/blob/master/bashlex/parser.py>

4.3.2.3 Bashast

<https://github.com/neloe/libbash/blob/master/bashast/bashast.g>

4.3.3 Spouštění příkazů

Popsat základní principy jak funguje shell. Popsat procesy v unixu, `fork`, `exec`, co všechno se musí stát, aby shell mohl spustit příkaz.

4.3.4 Struktura BASHe

todo: Zdrojový kód je rozdělen do souboru, možná by bylo dobře popsat popsat co který soubor dělá, aby si čtenář udelal alespoň trochu obrázek.

4.4 Debugování shellu

tohle je jeden bod zadání: "Proveďte řešení existujících nástrojů pro statickou analýzu, krokování a hledání chyb v BASH skriptech."

4.4.1 Debugování Bashe

todo: návod na debugování bashe přímo pomocí `gdb`. (tohle nakonec pro moji práci nebylo potřeba)

4.4.2 Interní nástroje

todo: popsat to, jak nástroje má v sobě `bash` zabudované v základu

4.4.2.1 Debugovací mód BASHe (jak funguje `shopt` s `extdebug`)

`shopt` s `extdebug`

4.4.2.2 set x, u, v, e

příklady do skriptu

4.4.2.3 PS0, PS4

PS0 bude v novém bashi, my ji proto nebudeme používat, PS4 se vypisuje při debugování. todo: ps0 se nakonec nepovedlo

4.4.3 Externí nástroje

4.4.4 BASH Debugger

todo: popsát jak funguje, co všechno umí, nějaké příklady <http://bashdb.sourceforge.net/>

4.4.5 BashEclipse

BashEclipse je plugin do Eclipse, který umí krokovat v gui eclipse.

<http://unix.stackexchange.com/questions/131491/is-there-a-gui-debugger-for-shell-scripts> <https://sourceforge.net/projects/shelled/>
<https://sourceforge.net/projects/basheclipse/>

4.5 Možnosti debugování v interaktivním shellu

Nebyl nalezen žádný nástroj pro debugování interaktivního shellu. todo

4.5.1 Definice debugování interaktivního shellu

todo: nemá cenu debugovat jednoduché příkazy, ale složité ano. chceme umět rozkrokovat pipy a subshelly (tedy to co nám to dělá teď)

4.5.2 GNU Readline

GNU Readline umožňuje přemapovat enter tak, abysme mohli spustit příkaz v naší definované funkci. Problémem je, že takto upravený příkaz se uloží do historie. Dalším problémem jsou víceřádkové příkazy, tedy takové, pro jejichž napsání musíme několikrát zmáchnout enter. TODO: ukázka.

4.5.3 Napsání nového REPLu

Zprovoznění základní funkcionality by bylo snadné, vzhledem ke komplexnosti BASHe však téměř nemožné mít stejné chování jako v BASHi.

4.5.4 DEBUG trap

Současné řešení. Při zapnutém extdebug je možné příkazy nepustit a jen evalovat poslední příkaz z historie. TODO: je potřeba popsat základní chování historie (např mezera na začátku příkazu, atd.)

4.6 Logování výstupu

Napsat o tom, jak a proč logovat výstup skriptu. Jak to dělat ve skriptech, jak to dělat v interaktivním shellu. Možnosti zapínání debugování ve skriptech. Popsat příkaz jak příkaz script, tak i logování přes exec. Popsat co se tam všechno děje.

todo: tohle jsem zkoušel a nebylo to úplně jednoduché, vzhledem k tomu, že to není v zadání bych to možná úplně vynechal

4.7 Automatizované spouštění příkazů

Popsat jak se dají automaticky spouštět příkazy. At už lokálně, nebo vzdalene. Popsat jak rekonstrukci z typescriptu, tak třeba Tcl, Expect.

4.7.1 Testování shell skriptů

Popsat jak fungují některé testovací frameworky. shunit2, roundup

4.8 Chování uživatelů v příkazové řádce

todo: tohle je asi úplně zbytečné. podle mě úplně stačí to, že frameworky mají vysoký počet instalací, takže se tyto věci používají

"Napadlo mě udělat anketu o tom, jak se uživatelé chovají v příkazové řádce. Například jaky používají shell, jaky mají PS1, nastavenou historii, logování atd, jaky používají aliasy, jestli si dělají skripty na každou věc, atd. Asi by stálo za to vysledovat chování studentu předmětu PS1 a pomoci jim v prevenci chyb které dělají."

4.9 Statická analýza skriptů

bash má validaci syntaxe: bash -n, ale to nezjistí skoro žádné chyby.

4.9.1 Check Bashisms

<http://checkbashisms.sourceforge.net/> Některé skripty cheme kvůli rychlosti spouštět v Bourne-Shellu. Protože ten je pouze podmnožina bashe, tento nástroj najde výskyty syntaxe, která je podporovaná pouze v bashi a v Bourne-Shellu nemusí fungovat správně.

4.9.2 Explain Shell

Tato analýza nehledá chyby, ale vyparsuje vložený kód a vyhledá příslušnou dokumentaci v manuálových stránkách. Využívá knihovny bashlex, kterou využijeme v našem debuggeru. <http://www.explainshell.com/>

4.9.3 ShellCheck

todo: tady bych se mohl rozepsat víc. o tom, že to je napsané v haskellu, o tom že existuje databáze věcí, které se ve skriptech hledají a že ke každým je wiki..

Realizace

v zadání je: Navrhněte a implementujte nástroj, který umožní psát uživatelské skripty pro analýzu příkazů a ovlivňování jejich spouštění a vykonávání. = to jsou hooky

Nástroj musí umožňovat krokovat složitější skripty po jednotlivých příkazech. = to je debugger

Pro analýzu spouštěných skriptů využijte vhodný nástroj z řešeršní části. = to je shellcheck

5.1 Nespouštění příkazů

Pro zabránění spouštění používáme DEBUG trap. todo: popsat problémy a řešení

5.2 Implementace debuggeru

todo: napsat o tom, že část je napsána v pythonu a že se celá věc spouští v pre-hooku, takže to je přesně ten čas, kdy se může příkaz debuggovat a originální příkaz se nespustí.

popsat parsování v bashlex a jak se celá věc předává do bashu

5.3 Hooks

Aby byl kód přehledný, kromě bezpečných příkazů je funkcionality rozdělena do hooků, nebo-li modulů, které obsahují kód, který je spuštěn před, nebo i po vykonání příkazu. Kód vykonaný před příkazem může rozhodnout, zda-li má dojít k zabránění vykonání příkazu.

5.3.1 Implementace hooků

todo: napsat o tom, že se hooky pouští automaticky před i po a že je snadné vytvořit vlastní hook

5.3.2 Hooky před vykonáním příkazu

todo: napsat co všechno můžu dělat

5.3.3 Hooky po vykonání příkazu

todo: napsat co všechno můžu dělat

5.3.4 Předvytvořené hooky

todo: napsat o tom, že

5.3.5 Shellcheck hook

todo: tohle je nejlepší věc, kterou celé commash umí. tady by to možná chtělo ukázat případy, kdy se to hodí. taky popsat to, že některé věci jdou snadno zakázat

5.3.6 Bashlex hook

todo: tohle je bashovská část debuggeru

5.3.7 Explain RC hook

5.3.8 Notfound hook

5.4 Bezpečný mód

Bezpečný mód umožňuje dvě základní věci. Tou jednodušší je pouze vypnutí efektu příkazu, který má nějaké destruktivní následky. Složitější varianta dovoluje vrácení do stavu před vykonáním příkazu.

todo: napsat o tom, že po zkoušení jsem došel k závěru, že je lepší uživateli nejprve ukázat co se děje, co se stane a jaké má možnosti. častokrát už

5.4.1 Bezpečné rm

todo: napsat o tom, jak to je vyřešené a proč. napsat, že `rm -i` i `rm -I` je k ničemu a proč je naše verze lepší popsat, jak funguje freedesktop.org trash a že ukládáme seznam souborů, které byly smazány jedním `rm`, aby se celý spuštěný příkaz `rm` vrátil

5.5 Historie

Popsat jak jsem vyresil ukladani historie prikazu, ukladani vystupu, navratove kody, jak vracet nasledky prikazu do puvodniho stavu. todo: historie taky neni v zadani a moc jsem to nedoresil. teoreticky by na to sel napsat pre a post hook, ktery zaznamena vsechno potrebne a zapise to nekam do souboru.

Závěr

todo: urcite bych napsal. ze nikdo nic podobneho neudelal a ohlasy byly celkem pozitivni. pokud se toho chytne alespon par lidi a budou hlasit chyby, co je potreba vylepsit, nebo dokonce i prispivat, mohl by z toho byt jednou opravdu vyborny pomocnik

sem napište závěr Vaší práce

Literatura

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

Obsah přiloženého CD

	readme.txt.....	stručný popis obsahu CD
	exe	adresář se spustitelnou formou implementace
	src	
	impl.....	zdrojové kódy implementace
	thesis	zdrojová forma práce ve formátu L ^A T _E X
	text	text práce
	thesis.pdf	text práce ve formátu PDF
	thesis.ps	text práce ve formátu PS