

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

30. června 2016



---

## 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 30. června 2016

.....

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2016 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, 2016. 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

<b>Úvod</b>	<b>1</b>
<b>1 Cíl práce</b>	<b>3</b>
<b>2 Analýza a návrh</b>	<b>5</b>
2.1 Historie UNIXu . . . . .	5
2.2 Historie shellu . . . . .	5
2.3 Fungování shellu . . . . .	5
2.4 Debugování shellu . . . . .	6
2.5 Možnosti debugování v interaktivním shellu . . . . .	6
2.6 Logování výstupu . . . . .	7
2.7 Automatizované spouštění příkazů . . . . .	7
2.8 Požadavky na interaktivní debugger . . . . .	7
2.9 Chování uživatelů v příkazové řádce . . . . .	7
<b>3 Realizace</b>	<b>9</b>
3.1 Nespouštění příkazů . . . . .	9
3.2 Hooks . . . . .	9
3.3 Bezpečný mód . . . . .	9
3.4 Historie . . . . .	9
<b>Závěr</b>	<b>11</b>
<b>Literatura</b>	<b>13</b>
<b>A Seznam použitých zkratk</b>	<b>15</b>
<b>B Obsah přiloženého CD</b>	<b>17</b>



---

## Seznam obrázků

2.1	Historie Shellu . . . . .	5
-----	---------------------------	---



---

# 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





## **Cíl práce**

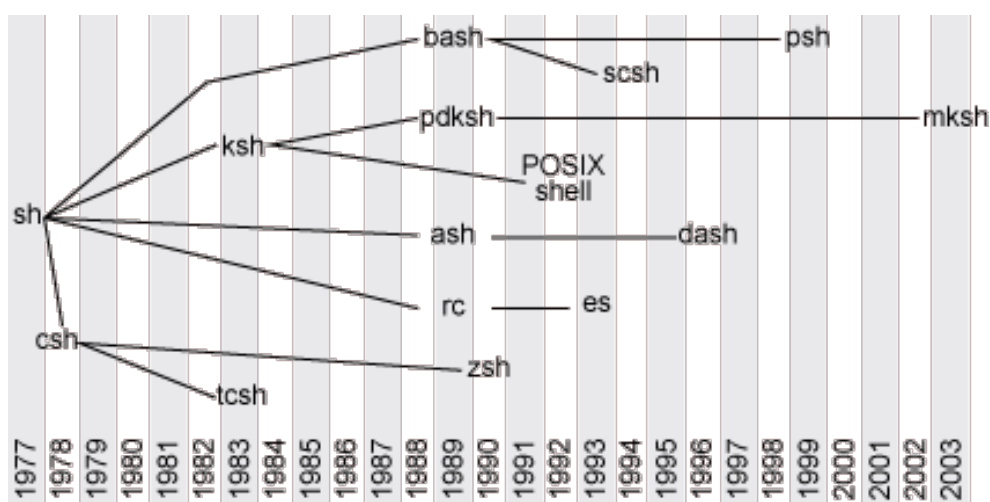
Napsat o tom, že cílem práce je usnadnit práci v příkazové řádce a sepsat základní funkcionalitu debuggeru.



## Analýza a návrh

### 2.1 Historie UNIXu

### 2.2 Historie shellu



Obrázek 2.1: Historie Shellu

V latexových komentářích jsou nějaké odkazy o historii.

### 2.3 Fungování shellu

#### 2.3.1 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 zatím tak složité není).

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

### 2.3.2 Spouštění příkazů

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

### 2.3.3 Struktura BASHe

Zdrojový kód je rozdělen do souboru, možná by bylo dobře popsat popsat co který soubor dělá, aby si čtenář udelel.

## 2.4 Debugování shellu

### 2.4.1 Debugování BASHe

Jestli jsem se k tomu nedostal, ale chtěl bych si odkrokovat v gdb pár základních věcí, abych viděl co se tam děje. Možná by stálo za to septat i sem.

### 2.4.2 Interní nástroje

#### 2.4.2.1 Debugovací mód BASHe (jak funguje shopt s extdebug)

shopt s extdebug

#### 2.4.2.2 set x, u, v, e

příklady do skriptu

#### 2.4.2.3 PS0, PS4

PS0 bude v novém bashi, my ji proto nebudeme používat, PS4 se vypisuje při debugování

### 2.4.3 Externí nástroje

### 2.4.4 BASH Debugger

todo: popsat jak funguje, co všechno umí, nějaké příklady

## 2.5 Možnosti debugování v interaktivním shellu

### 2.5.1 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.

### 2.5.2 Napsání nového REPLu

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

### 2.5.3 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.)

## 2.6 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.

## 2.7 Automatizované spouštění příkazů

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

### 2.7.1 Testování shell skriptů

Popsat jak fungují některé testovací frameworky. shunit2, roundup

## 2.8 Požadavky na interaktivní debugger

Zde by mohly být sepsány obecné požadavky na to, co by interaktivní debugger měl vlastně dělat.

## 2.9 Chování uživatelů v příkazové řádce

Napadlo mě udelat aketu 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 stalo za to vysledovat chování studentů předmětu PS1 a pomoci jim v prevenci chyb které dělají.



---

## Realizace

### 3.1 Nespouštění příkazů

Pro zabránění spouštění používáme DEBUG trap.

### 3.2 Hooks

Aby byl kód přehledný, veškerá funkcionalita je 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.

### 3.3 Bezpečný mód

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

### 3.4 Historie

Popsat jak jsem vyřel ukladani historie prikazu, ukladani vystupu, navratove kody, jak vracet nasledky prikazu do puvodniho stavu.





---

# **Závěr**

sem napište závěr Vaší práce



---

## Literatura



## Seznam použitých zkratk

**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 <sup>A</sup> T <sub>E</sub> X
	text .....	text práce
	thesis.pdf .....	text práce ve formátu PDF
	thesis.ps .....	text práce ve formátu PS