A7B38UOS Úvod do operačních systémů

2. Cvičení

Seznámení se shellem, textové editory numerické výpočty

Obsah cvičení

Práce s proměnnými

Zpracování příkazové řádky

Prompt

Historie

Editor vi

Příkazy - ls

Slouží k výpisu obsahu adresáře

Syntaxe

ls [přepínače] [adresář]

Přepínače

bližší informace na dalším cvičení

Př.

ls -l ./pokus

Příkazy - sleep

Slouží k uspání vstupu na určitý čas (v sekundách), do konzole je možné psát, ale příkazy se vykonají až po uplynutí nastaveného času

Syntaxe

sleep time

Př.

sleep 20

Příkazy - cat

Slouží k sloučení souborů a zobrazení na standardní výstup

Syntaxe

cat [přepínače] [soubor]

Přepínače

n	zobrazí čísla řádků
b	zobrazí čísla neprázdných řádků
T	zobrazí tabelátory jako ^I
E	zobrazí $ na konci řádku

Př. zobrazení obsahu souboru text.txt

cat text.txt

Příkazy - wc

Slouží k zobrazení počtu bytů, znaků, slov a řádek vstupu

Syntaxe

wc [přepínače] [soubor]

Přepínače

m	zobrazí počet znaků
l	zobrazí počet řádků
L	zobrazí znaků nejdelšího řádku (jeho délku)
w	zobrazí počet slov

Př.

wc -l text.txt

Práce s proměnnými

Proměnné jsou „lokální“ a „globální“

- lokální proměnné se nedědí do procesů potomků
- globální proměnné se dědí do procesů potomků (prostředí)

Změna lokální proměnné na globální příkazem *export*

Proměnná má platnost do jejího zrušení nebo do ukončení shellu

Proces potomka nemůže změnit hodnotu proměnné v rodičovském procesu

Práce s proměnnými

Přiřazení hodnoty proměnné

jméno_proměnné=hodnota

Pozn. proměnné nemají datový typ, jsou brány jako řetězce

Př.

mojedata=10

Použití hodnoty proměnné

$*jméno_proměnné*

Př. - výpis hodnoty proměnné

echo $mojedata

Práce s proměnnými

Výpis všech proměnných

set

Výpis nastavení prostředí

env

Smazání proměnné

unset jméno_proměnné

Export proměnné (změna z lokální na globální)

export jméno_proměnné

Zpracování příkazové řádky

Znaky rušící význam speciálních znaků

" " slabé rušení významu speciálních znaků v řetězci

' ' silné rušení významu speciálních znaků v řetězci

\ rušení významu následujícího speciálního znaku

Pozn. někdy je nutné zkombinovat slabé i silné rušení významu speciálních znaků

Zpracování příkazové řádky

Komentáře

Znakem # začínají komentáře do konce řádku

Lze je používat i v příkazovém řádku pro komentování aktuální práce

Tento znak se dále používá ve skriptech pro jejich uvození viz cvičení 10

Pozn. Doporučujeme při práci komentáře používat, nelze spoléhat na to, že si po čase vzpomenete, jakého výsledku jste chtěli dosáhnout.

Zpracování příkazové řádky

Slučování příkazů na příkazové řádce

V příkazové řádce se běžně používá více příkazů zároveň, umožňuje to zpracovávat data bez ukládání do dočasných souboru na disk

Pokud je potřeba použít složitější konstrukci je nutné jednotlivé příkazy uzavírat mezi závorky

Sekvenční vykonávání příkazů bez předávání dat mezi nimi

Př.

mojedata=10; echo $mojedata

Zpracování příkazové řádky

Slučování příkazů na příkazové řádce

Sekvenční vykonávání příkazu s předáním dat - výstup prvního příkazu je vstupem druhého příkazu

Př.

date | *wc -m*

Paralelní spouštění příkazů – první z nich běží na popředí, druhý na pozadí

Př.

cat text.txt&(sleep 20;echo "cas vyprsel")

Zpracování příkazové řádky

Logické slučování příkazů na příkazové řádce

Zkráceně vyhodnocený logický AND - druhý příkaz bude spuštěn pouze pokud první **neskončí** chybou

Př.

cat text.txt && echo "v poradku"

Zkráceně vyhodnocený logický OR - druhý příkaz bude spuštěn pouze pokud první **skončí** chybou

Př.

cat text1.txt || *echo "nastala chyba"*

Zpracování příkazové řádky

Rozdělení na slova

$IFS proměnná určující oddělovač slov

Standardními oddělovači slov jsou :space, tab, newline

Při práci s různými soubory (např./etc/passwd) je vhodnější používat jiné oddělovače slov (např. „:“)

Při změně je nutné dát **pozor**, změna **má vliv na činnost mnoha příkazů**

Měňte jen, když opravdu víte, co činíte. A jakmile změněnou hodnotu **již nepotřebujete, obnovte původní stav**.

Zpracování příkazové řádky

Přesměrování

Přesměrování se používá k změně standardního vstupu nebo výstupu

Používají se znaky <, <<, >, >>, <&, >&

Bližší informace viz. cvičení 4

Př. přesměrování st. výstupu do souboru text.txt

ls -l >text.txt

Zpracování příkazové řádky

Náhrada jmen souborů

Při práci se soubory je možné nahrazovat znaky/řetězce pomocí speciálních znaků

?	náhrada jednoho znaku
*	náhrada řetězce znaků
[rozsah]	náhrada pro znaky v uvedeném rozsahu
[^rozsah]	náhrada pro znaky mimo uvedený rozsah

Př. výpis všech souboru jejichž jméno má dva znaky a prvním znakem je „s“

cat s?

Zpracování příkazové řádky

Hledání příkazů

Pro případné doplňování příkazů lze jako nápovědy používat TAB - zobrazí se možné příkazy obsahující již napsané znaky

Pro vyhledávání příkazů lze používat příkazy *whereis* nebo *which*

which	slouží k hledání cesty k příkazu
whereis	slouží k hledání cesty ke zdroji a manuálovým stránkám příkazu (je komplexnější než *which*)

Zpracování příkazové řádky

Expanze {}

Znaky uvedené uvnitř { } se nahrazují vždy společně – aplikují se všechny do daného řetězce

Př.

cat d{1,2,3,4}.txt

spojí a vypíše soubory d1, d2, d3 a d4

lze zapsat také jako *cat d1 d2 d3 d4*

Zpracování příkazové řádky

Expanze ~

Znak ~ nahrazuje cestu k domovskému(home) adresáři uživatele

V případě, že za znakem ~ následuje uživatelské jméno, nahrazuje cestu k domovskému(home) adresáři daného uživatele

Př.

ls ~

Zpracování příkazové řádky

Expanze parametru / proměnné

Složené závorky umožňují bezpečně odlišit název proměnné od ostatního textu.

Př. výpis proměnných PATH,0 (nultý parametr),USER2

echo $PATH $0 ${USER}2

Př. přidání dalšího obsahu k obsahu proměnné

PATH=$PATH:$HOME/scripts

Zpracování příkazové řádky

Náhrada příkazu

V případě že uvedeme příkaz mezi `` (zpětné apostrofy) je provedena náhrada výsledku příkazu jako text v uvedeném místě. Dnes se však spíše používá $(příkaz) se stejným významem.

Př.

mojedata=`date`

proměnná mojedata obsahuje výsledek příkazu *date*

Zpracování příkazové řádky

Numerické výpočty

Při zpracování je možné používat jednoduché numerické výpočty

Vlastní výpočet je uzavřen mezi (()) a použit jako proměnná

Př. uspání vstupu na 50 sekund

*sleep $((5*10))*

Zpracování příkazové řádky

Word splitting

Výpisy lze zpracovávat i s odstraněním příslušného formátování (tabelátory, konce řádků)

Porovnejte následující dva výpisy:

ls -l

echo `ls -l`

První z výstupů je formátován, druhý již ne

Prompt +

Proměnné PS1 až PS4

- PS1 primární prompt
- PS2 sekundární prompt (pokračování na nové řádce)
- PS3 výzva pro výběr (příkaz select)
- PS4 znak při trasování programu (set -x)

Některé speciální sekvence

- \d datum (den měsíc rok)
- \t čas (HH:MM:SS) , případně \A pro čas (HH:MM)
- \h , \u jméno počítače (hostname), jméno uživatele (username)
- \w aktuální adresář (working directory)
- \! pořadí v historii , případně \# pro pořadí příkazu v akt. shellu
- \$ znak $ pro uživatele a znak # pro root

Historie (příkazů)

Slouží prohlížení použitých příkazů

Syntaxe

history

Pro použití příkazu z historie se užívá zápis pomocí !

!n	spustí n-tý příkaz z historie (od začátku)
!-n	spustí poslední n-tý příkaz z historie (od konce)
!!	spustí poslední příkaz (předchozí)
!string	spustí poslední příkaz začínající na řetězec string
!?str?	spustí poslední příkaz obsahující řetězec str
!^str1^str2^	spustí poslední příkaz a nahradí řetězec str1 za str2

Textové editory

Na tomto systému je nainstalováno několik textových editorů (editorů pracujících s „čistým“ textem).

vi (vim)

textový režim, funguje na všech terminálech

joe

textový režim, ovládání klávesovými zkratkami, nápověda ^KH

gedit

grafický režim

Editor vi

Textový editor (Visual editor) – program vi

Dostupný na většině platforem a OS

Nepotřebuje žádné speciální klávesy kromě ESC

Vylepšená verze – program vim (Vi IMproved)

Syntaxe:

vi (soubor)

Režimy editoru

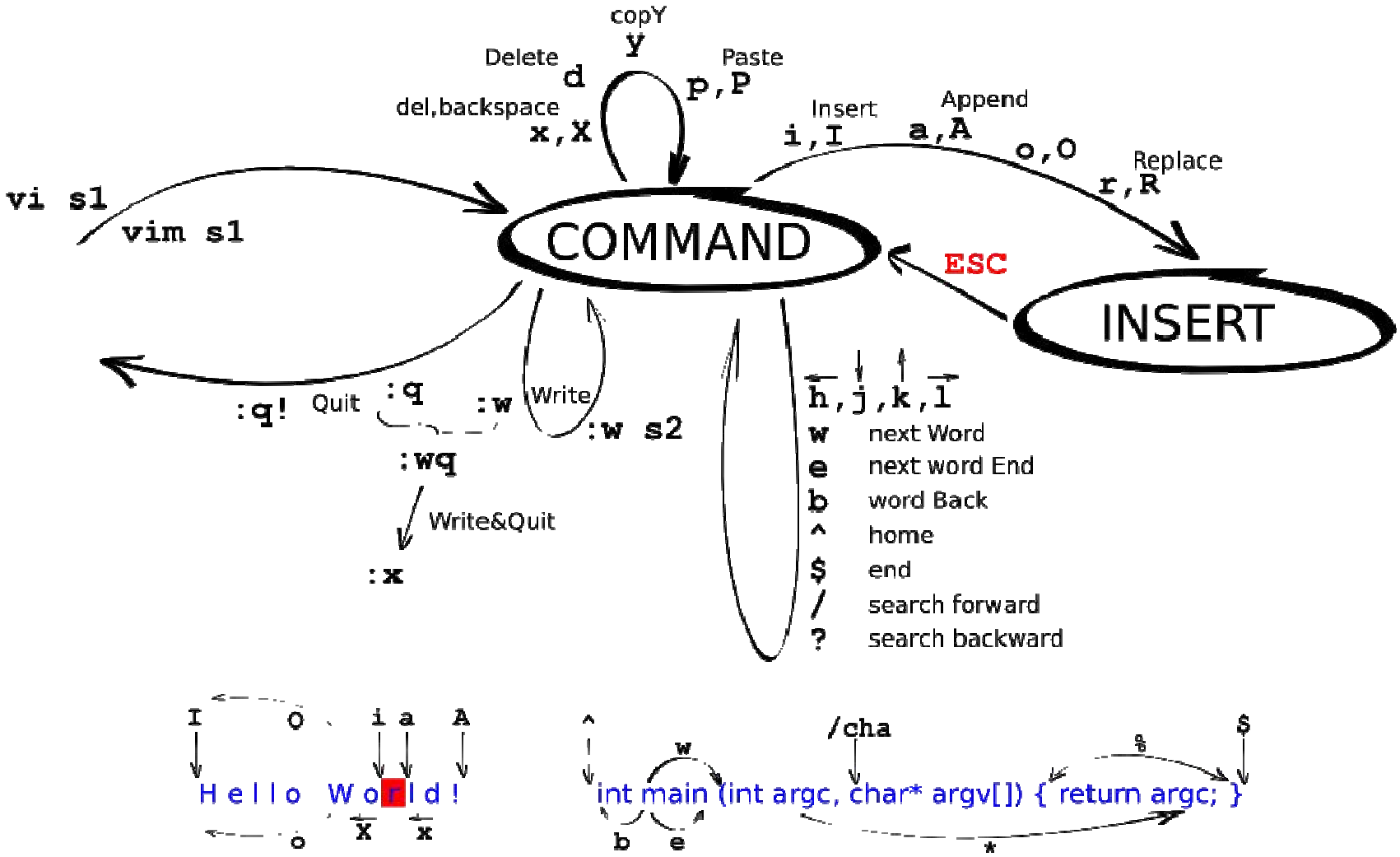
Příkazový (ESC)

Vkládací

Základní příkazy

Ukončení editoru	:q nebo :q!
Uložení	:w
Pohyb	hjkl
Vložení textu	i
Smazání znaku	x
Smazání řádky	dd

Editor vi - příkazy +



Editor vi - příkazy +

Značka v textu (mark)	mx
Skok na značku	`x
Skok na předchozí místo	``
Skok na konec souboru	G
Skok na řádku 5	5G
Spuštění příkazu v shellu	:! ls
Vložení výstupu příkazu	:r! date
Náhrada slova (prvního výskytu)	:s/slovo/jine/
Náhrada slova na celé řádce	:s/slovo/jine/g
Náhrada slova v celém souboru	:%s/slovo/jine/g