Teorie jazyků a automatů

Regulární výrazy v praxi

Šárka Vavrečková

Ústav informatiky
Filozoficko-Přírodovědecká fakulta
Slezské univerzity, Opava

31. března 2008

Windows – `dir`

Určení příkazu

Výpis obsahu složky (adresáře)

Syntaxe (silně zkrácená)

```
dir [Jednotka:][Cesta][Název_souboru] [...] parametry
```

Regulární výrazy pro `dir`

Regulární výrazy mohou být využity při stanovení názvu souboru a cesty k němu.

Prvek	**Význam**
?	Libovolný znak (právě jeden)
*	Nula nebo více výskytů jakýchkoliv znaků, řetězec

Windows – `dir`

Příklady

- `dir c:\win*`
  hledá přímo na disku `C:` vše, co začíná `win`
- `dir nt????*.txt /s /b`
  hledá soubory s příponou `txt` začínající `nt`, následují jakékoliv 4 znaky, parametry za řetězcem způsobí rekurzívní prohledávání všech podadresářů

Podobné jednoduché regulární výrazy se používají i u dalších příkazů v Příkazovém řádku.

Windows – `findstr`

Určení příkazu

Vyhledávání řetězců v souborech

Windows – `findstr`

Syntaxe (silně zkrácená)

```
findstr [/b] [/e] [/i] [/n] [/p] [/g:soubor] [/f:soubor]
[/c:řetězec] [řetězce] [Název_souboru [...]]
```

- `/b` hledá shodu na začátku řádku
- `/e` hledá shodu na konci řádku
- `/i` nerozlišuje malá a velká písmena
- `/n` zobrazí také čísla řádků, na kterých byla nalezena shoda
- `/p` vynechá soubory obsahující netisknutelné znaky
- `/g:soubor` z tohoto souboru čte řetězce s reg. výrazy
- `/f:soubor` z tohoto souboru čte názvy souborů, ve kterých má vyhledávat
- `/c:řetězec` řetězec bere jako jediný regulární výraz
- `řetězce` regulární výrazy pro vyhledávání oddělené mezerami

Windows – `findstr`

Regulární výrazy ve `findstr`

Prvek	**Význam**
`.`	Libovolný znak
`*`	Nula nebo více výskytů předcházejícího znaku nebo třídy
`^`	Začátek řádku
`$`	Konec řádku
`[třída]`	Jakýkoli (jeden) znak z množiny v závorkách
`[^třída]`	Jakýkoli znak mimo prvky množiny
`[x-y]`	Jakékoli znaky v daném rozsahu (jeden)
`\<xyz`	Začátek slova
`xyz\>`	Konec slova

Windows – `findstr`

Příklady

- `findstr "for med:=1" hlavni.pas`
  hledá řetězce `for` a `med:=1` v souboru `hlavni.pas`
- `findstr /c:"for med:=1" hlavni.pas`
  hledá řetězec `for med:=1` v souboru `hlavni.pas`
- `findstr /i "^.boot" c:\boot.ini`
  hledá v souboru `C:\boot.ini` řetězec `boot`, který začíná od druhého písmene na řádku (symbol `^` značí začátek řádku, za ním tečka určuje jakýkoliv symbol)
- `findstr /i /c:"^ *for([ijk]" *.c`
  hledá v souborech s příponou `.c` řádky, na jejichž začátku (po libovolném počtu mezer) je klíčové slovo `for`, závorka a pak některé z písmen `i`, `j` nebo `k`

Linux – grep

Určení příkazu

Vyhledávání řetězců v souborech

Syntaxe (silně zkrácená)

`grep [přepínače] reg_výraz [soubor ...]`

- `-i` nerozlišuje malá a velká písmena
- `-l` pouze vypíše názvy souborů, ve kterých nalezl shodu
- `-n` vypíše také číslo řádku
- `-r` rekurzívně zpracovává i podadresáře
- `-o` vypíše jen nalezený řetězec, ne celý řádek
- `-c` v souborech pouze spočítá výskyty nalezeného řetězce

Linux – grep

Regulární výrazy ve grep

Prvek	Význam
`.`	Libovolný znak
`?`	Nula nebo jeden výskyt předcházejícího řetězce
`*`	Nula nebo více výskytů předcházejícího řetězce
`+`	Jeden nebo více výskytů předcházejícího řetězce
`{m}`	`m` opakování předcházejícího řetězce
`{m,n}`	`m` až `n` opakování předcházejícího řetězce
`{m,}`	`m` nebo více opakování předcházejícího řetězce
`^`	Začátek řádku
`$`	Konec řádku
`[třída]`	Jakýkoli (jeden) znak z množiny v závorkách
`[^třída]`	Jakýkoli znak mimo prvky množiny
`[x-y]`	Jakékoli znaky v daném rozsahu (jeden)
`r1 | r2`	logické nebo – buď řetězec `r1`, nebo řetězec `r2`

Linux – grep

Příklady

- `grep -i "vypis" *.txt`
  hledá v souborech s příponou `.txt` slovo `vypis` bez rozlišování malých a velkých písmen
- `grep -il "vypis" *.txt`
  hledá v souborech s příponou `.txt` slovo `vypis` bez rozlišování malých a velkých písmen, pouze vypíše názvy souborů, ve kterých se toto slovo nachází
- `grep -cr "#include.*\.[hc]" *.c`
  u každého souboru s příponou `.c` vypíše počet vkládaných souborů s příponou `.h` nebo `.c`, a protože tečka je významový znak (určuje jeden jakýkoliv symbol), musíme před ni dát zpětné lomítko, aby byla brána jako součást řetězce
- `grep -cr "Dear \(Mr.|Ms.|Miss\)" *.txt`

Linux – grep

Posix sekvence

Můžeme napsat 0-9, a nebo `[:digit:]`. Další možnosti:

`[:digit:]`	číslice
`[:xdigit:]`	hexadecimální číslice
`[:alpha:]`	písmena
`[:alnum:]`	písmena a číslice
`[:lower:]`	malá písmena
`[:upper:]`	velká písmena
`[:blank:]`	mezera a tabulátor
`[:space:]`	prázdné znaky (mezera, tabulátor, konec řádku, ...)
`[:graph:]`	viditelné znaky
`[:print:]`	viditelné znaky a mezera

Linux – grep

Příklad

Hledáme rodná čísla ve tvaru 123456/1234 – nejdřív 6 číslic, pak lomítko a tři nebo čtyři číslice

```
grep '[0-9]\{6\}/[0-9]\{3,4\}' soubor.txt
```

nebo

```
grep '[[:digit:]]\{6\}/[[:digit:]]\{3,4\}' soubor.txt
```

Příklad

Hledáme IP adresy ve tvaru 123.123.123.123 – čtyři skupiny číslic, v každé skupině 1 až 3 číslice:

```
grep '\([[:digit:]]\{1,3\}\)\{4\}' soubor.log
```

je totéž jako

```
grep '\([0-9]\{1,3\}\)\{4\}' soubor.log
```

Linux – sed

Určení příkazu

Zpracovávání výrazů v souborech

Syntaxe (silně zkrácená)

`sed [přepínače]... [script] [vstupní_soubor]...`

- `-e řetězec` skript k provedení (ne soubor!)
- `-f soubor` soubor se skriptem k provedení

Skripty

Skript má formu `ap`, kde `a` je adresa ve zpracovávaném souboru a `p` je příkaz, který se na daném místě má provést.

Linux – sed

Adresy

- (bez adresy) příkaz se použije na všechny řádky
- `číslo` číslo řádku, který se má zpracovat
- `číslo~krok` všechny řádky od řádku s daným číslem s násobkem daným krokem (tj. `číslo + i*krok`), například `1~2` budou všechny liché řádky
- `$` poslední řádek vstupu
- `/regulární_výraz/` řádky odpovídající regulárnímu výrazu, lomítka jsou nutná, uvnitř používáme vše, co u grep, ale před některé významové znaky dáváme `\` (`\+`, `\?`, `\{...}`, `\|`) Pokud následuje `I`, nerozlišují se malá a velká písmena
- `adresa1,adresa2` všechny řádky v rozmezí adres
- `adresa,+n` všech n řádků od zadané adresy

Linux – sed

Příkazy

- `d` vymaž nalezený vzor
- `p` vypiš (vytiskni) nalezený vzor na výstup
- `s/co/čím/přepínače` (lomítka jsou součástí výrazu) nahraď `co` `čím`, a to způsobem určeným přepínači:
  - `g` nahraď všechny výskyty
  - `číslo` nahraď jen výskyt s pořadím `číslo`
  - `i` nerozlišuj malá a velká písmena

Linux – sed

Příklady

- `sed -e '1,5d' soubor.txt`
  odstraní ze zadaného souboru první až pátý řádek
- `sed -e 's/<[^>]*>/g' *.html`
  ze všech html souborů v daném adresáři odstraní všechny tagy `<...>` (vnitřní část zajišťuje, že pokud je na řádku více tagů, bude text mezi nimi zachován), dvě lomítka za sebou jsou nutná, protože mezi nimi se vlastně nachází prázdný řetězec čím – g znamená „nahradit všechny výskyty“
- `cat soubor.txt | sed -e 's/\&/\&/g' | sed -e 's/</\</g' | sed -e 's/>/\>/g' > soubor.html`
  usnadní převod textového souboru do html formy tím, že všechny symboly & nahradí patřičným ekvivalentem v HTML kódu, totéž udělá také se znaménky < a >

Linux – sed

Paměť

Nalezené části výrazu jsou zapamatovány a přístupny pod `\1`, `\2`, atd., ukážeme si na příkladu dávkové změny přípony HTM na HTML:

```
ls *.htm | sed -e 's/\(.*\)\.htm/mv \1.htm \1.html/'
```

- příkaz `ls` vypíše obsah adresáře (jako `dir` ve Windows) podle zadaného vzoru – všechny soubory s příponou `htm`,
- ve skriptu použijeme příkaz `s/co/čím/přepínače`,
- reg. výraz pro „co“ znamená nějaké znaky (tj. `\(.*\)`) následované tečkou (`\.`) a původní příponou,
- provede se volání příkazu `mv`, který přejmenuje soubory s příponou HTM na tytéž soubory, ale s příponou HTML (část před příponou je zapamatována v `\1`).

Windows

Možnosti využití regulárních výrazů:

- jednoduché regulární výrazy (`*`, `?`) – prakticky ve všech příkazech, které pracují se složkami (adresáři) a soubory
- složitější regulární výrazy – pouze `findstr` nebo nástroje „třetích stran“

Informace

Nápověda Windows, klíčové slovo `findstr`.

Linux

Možnosti využití regulárních výrazů:

- jednoduché regulární výrazy (širší, vč. `[xyz]`, `[^xyz]` apod.) ve všech příkazech pracujících s adresáři a soubory
- grep – vyhledávání řetězců podle zadaného regulárního výrazu
- sed – vyhledávání a editace (různé akce) řetězců nejen podle zadaného regulárního výrazu
- awk – ještě širší možnosti než sed, programovací jazyk (podobný C) na zpracování textových souborů

Linux

Informace

- Nápověda daného systému v grafickém režimu.
- Manuálové stránky, v textovém shellu nebo na internetu (třeba na `google.com`) zadáme k vyhledání `man grep`, `man sed` apod.
- Na internetu najdeme hodně příkladů, například `grep příklady`