Základy zpracování dat pomocí AWK

Tomáš Kühr

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Jazyk AWK

- vytvořen 1977 v Bellových laboratořích
- autoři Alfred Aho, Peter Weinberger a Brian Kernighan
- plnohodnotný programovací jazyk pro zpracování dat
- podporuje
  - vestavěné i vlastní proměnné
  - operátory
  - větvení (if-else)
  - cykly (for, while, do-while)
  - vestavěné i vlastní funkce
  - pole (i vícerozměrná)
  - práce s regulárními výrazy
- syntaxe vychází z jazyka C
- vstupním proudem je obvykle soubor nebo výstup jiného programu
- výstup může být uložen do souboru nebo použit jako vstup jiného programu

Zpracování souboru pomocí AWK

1. vyhodnocení BEGIN bloku
2. postupné zpracování všech řádků
   a. načte řádek vstupního proudu (souboru)
   b. postupně se prochází jednotlivé bloky příkazů
      i. zjistí se, zda se má daný blok příkazů na načtený řádek použít
      ii. pokud ano, spustí se blok příkazů
      iii. přejde se k dalšímu bloku příkazů
   c. přejde se k dalšímu řádku
3. vyhodnocení END bloku

Možnosti spuštění AWK

Příkazy zadané přímo do promtu

- pokud je příkazů malé množství a jsou spíše jednodušší
- pokud si nechceme příkazy uložit pro pozdější použití
- příklad: `awk -e '{print}' -e '{print}' books.txt`

Příkazy zadané v souboru

- pokud je příkazů více nebo jsou složitější
- pokud chceme stejné příkazy používat někdy v budoucnu
- příklad: `awk -f commands.awk books.txt`
- pokud příkazy v souboru doplníme prvním řádkem: `#!/usr/bin/awk -f`
- a nastavíme právo spouštět daný soubor
- lze použít také `./script.awk books.txt`

Nejobvyklejší přepínače

- předání více příkazů z promtu (přepínač -e)
- zpracování příkazů ze souboru (přepínač -f)
- nastavení oddělovače sloupců (přepínač -F)
- předání proměnné do programu (přepínač -v)

Možnosti výběru řádků v AWK

Blok BEGIN

- příkazy se provedou před zpracováním prvního řádku
- může být použit k vypsání nějaké úvodní informace
- nebo k inicializaci proměnných
- příklad:
  `awk 'BEGIN {print "My books:"} {print}' books.txt`

Blok END

- příkazy se provedou po zpracování posledního řádku
- může být použit k vypsání nějaké závěrečné informace
- nebo k výpisu výsledku výpočtu, který závisí na všech položkách (např. součet, průměr, ...)
- příklad:
  `awk '{print} END {print "That\x27s all..."}' books.txt`

Možnosti výběru řádků v AWK

Blok aplikovaný na všechny řádky

- blok, před kterým není uvedena žádná specifikace
- nejběžnější a nejčastější varianta bloku
- lze kombinovat a příkazem if a tím omezit působnost příkazů libovolnou podmínkou

Blok aplikovaný na řádky odpovídající vzoru

- vzor píšeme před blok mezi znaky / /
- vzorem může být řetězec i regulární výraz
- pokud se na daném řádku vyskytuje jakýkoli (pod)řetězec odpovídající vzoru, příkazy se vykonají
- příklady:

```
awk '/Paulo/ {print}' books.txt
awk '/w/ {print}' books.txt
```

Regulární výrazy v AWK

- 1 libovolný znak (znak . ve vzoru)
- nepovinný výskyt znaku (? za daným znakem)
- libovolný (i nulový) počet opakování znaku (* za daným znakem)
- nenulový počet opakování znaku (+ za daným znakem)
- přesně daný počet opakování ({*počet*} za daným znakem)
- počet opakování v daném rozsahu ({*od*, *do*} za daným znakem)
- alespoň daný počet opakování ({*od*, } za daným znakem)
- libovolný znak z množiny ([*výčet*] nebo [*od-do*])
- libovolný znak mimo množinu ([^*výčet*] nebo [^*od-do*])
- lze použít i mnoho předdefinovaných skupin znaků ([:alpha:], [:digit:], ...)
- definování alternativ pomocí logického spojky „nebo“ (znaky |)
- možnost pracovat se začátky řádků (znak ^)
- možnost pracovat s konci řádků (znak $)
- seskupování znaků pro potřeby výše zmíněných operací pomocí (a)

Příklady použití regulárních výrazů v AWK

- `awk '/^The/ {print}' books.txt`
- `awk '/^.) The/ {print}' books.txt`
- `awk '/[468]$/ {print}' books.txt`
- `awk '/(Tolkien|Martin),/ {print}' books.txt`
- `awk '/[[:digit:]]{4}/ {print}' books.txt`
- `awk '/100?/ {print}' numbers.txt`
- `awk '/100?$/ {print}' numbers.txt`
- `awk '/10{5}$/ {print}' numbers.txt`
- `awk '/10{5,7}$/ {print}' numbers.txt`
- `awk '/10{5,}$/ {print}' numbers.txt`
- `awk '/1(00)*$/ {print}' numbers.txt`
- `awk '/1(00)+$/ {print}' numbers.txt`

Proměnné v AWK

- AWK pracuje s mnoha vestavěnými proměnnými
- tyto obsahují informace o vstupním souboru, aktuálním řádku i požadavcích na výstup

$0

- obsahuje celý aktuální řádek
- příklad: `awk '{print "["$0 "]"}' marks.txt`

$*n*

- obsahuje *n*-tou položku aktuálního řádku
- příklad: `awk '{print $3 "\t" $2}' marks.txt`

FILENAME

- obsahuje jméno vstupního souboru
- příklad: `awk 'END {print FILENAME}' marks.txt`

Proměnné v AWK

FS

- obsahuje informaci o oddělovači sloupců ve vstupním proudu
- implicitně libovolné množství bílých znaků, ale dá se změnit (přepínač -F)
- příklad: `awk -F : '{print $1} END {print FS}' marks.txt`

OFS

- oddělovač sloupců ve výstupním souboru
- implicitně mezera, ale dá se změnit
- příklad: `awk 'BEGIN {OFS = ", "} {print $2, $3}' marks.txt`

RS

- oddělovač položek ve vstupním souboru
- implicitně přechod na nový řádek

ORS

- oddělovač položek ve výstupním souboru
- implicitně přechod na nový řádek
- příklad: `awk 'BEGIN {RS = ""; ORS = ", "} {print}' marks.txt`

Proměnné v AWK

NF

- počet položek na aktuálním řádku
- příklad: `awk '{print $0, NF}' books.txt`

NR

- číslo aktuálně zpracovávaného řádku
- příklad: `awk '{print NR, $2, $4}' marks.txt`

Vlastní proměnné

- vytváříme a používáme podobně jako v jiných jazycích (např. Python)
- proměnnou lze nastavit už při spouštění AWK (přepínač -v)
- příklad:

```
awk -v count=10 '{count = count + 1} END {print count}' books.txt
```

Operátory v AWK

Aritmetické operátory

- s obvyklým významem +, -, *, /, % (zbytek po dělení)
- umocňování **, ^
- inkrementace ++ a dekrementace -- (zvýšení a snížení hodnoty proměnné o 1)

Operátory přiřazení

- základní operátor =
- modifikace proměnné +=, -=, *=, /=, %=, **=, ^=

Porovnávání

- pro porovnávání čísel i řetězců
- s obvyklým významem ==, !=, <, >, <=, >=

Logické operátory

- pro konstrukci složitějších podmínek
- spojky „a zároveň“ &&, „nebo“ || a negace !

Operátory v AWK

Spojení řetězců

- stačí uvést řetězce za sebou (oddělené mezerou)
- příklad: `spojeni = str1 str2`

Operátory pro práci s poli

- operátor indexu pro přístup k prvkům [ ]
- operátor testující, zda pole obsahuje daný prvek `in`

Operátory regulárních výrazů

- testují, zda první operand je obsahuje výraz zadaný druhým operátorem
- varianty ~ (je obsažen) a !~ (není obsažen)
- příklady:

```
awk '$1 ~ "100*" {print}' numbers.txt
awk '{if ($1 ~ "100*") print}' numbers.txt
```

Bodované úlohy

1. Implementujte v awk zjednodušenou verzi wc: výpis počtu znaků, slov (řetězců oddělených mezerami) a řádků v textu. Můžete použít funkci length pro zjištění délky řetězce. (1 bod)
2. Napište AWK skript, který všechny řádky začínající řetězcem „end: “ přesune na konec souboru a řetězec „end:“ při tom zcela vynechá. (2 body)