System operacyjny Linux

Paweł Rajba

pawel.rajba@continet.pl
http://kursy24.eu/

Zawartość modułu 7

- Język awk
  - Wprowadzenie
  - Schemat programu
  - Konstrukcja wzorców
  - Konstrukcja wyrażeń regularnych
  - Struktury kontrolne
  - Predefiniowane zmienne
  - Predefiniowane funkcje
  - Tworzenie własnych funkcji

Wprowadzenie

- Autorzy
  - Aho
  - Kernighan
  - Weinberger
- Myślenie w AWK
  - plik dzielimy na wiersze
  - wiersze dzielimy na pola
  - domyślnie:
    - wiersze są oddzielone końcami wierszy
    - pola są oddzielone białymi znakami

Schemat programu

- wzorzec1 { akcja1 }

  wzorzec2 { akcja2 }

  ...

  wzorzecN { akcjaN }

Konstrukcja wzorców

- Konstrukcja wzorców
  - BEGIN — blok tego wzorca będzie wykonywany przed rozpoczęciem przetwarzania pliku,
  - END — blok tego wzorca będzie wykonywany po zakończeniu przetwarzania pliku
  - /wyrażenie regularne/, np. /a.*a/
  - wyrażenie relacyjne, np. $1<3
  - wzorzec && wzorzec
  - wzorzec || wzorzec

Konstrukcja wzorców

- Konstrukcja wzorców c.d.
  - wzorzec ? wzorzec1 : wzorzec2
  - (wzorzec) — zgrupowanie,
  - ! wzorzec — zaprzeczenie, operator NOT, czyli wiersz nie może się dopasować do wzorzec,
  - wzorzec1, wzorzec2 — zakres tzn. po dopasowaniu się pewnego wiersza do wzorzec1 przetwarzane będą wszystkie kolejne wiersze bezwarunkowo, do momentu, aż których z tych wierszy nie dopasuje się do wzorzec2.

Konstrukcja wyrażeń regularnych

- Dopasowanie do
  - c; \c — znaku zwykłego; specjalnego,
  - . — dowolnego znaku, w tym znaku końca wiersza,
  - ^ — początku wiersza,
  - $ — końca wiersza,
  - [abc...] — dowolnego znaku z listy,
  - [^abc...] — dowolnego znaku nie będącego na liście,

Konstrukcja wyrażeń regularnych

- Dopasowanie do:
  - r1|r2 — r1 lub r2,
  - r1r2 — r1 i potem r2 (złożenie),
  - r+ — jednego lub więcej powtórzeń r,
  - r* — zero lub więcej powtórzeń r,
  - r? — zero lub jednego powtórzenia r,
  - (r) — do r (zgrupowanie),
  - r{n}; r{n,}; r{n,m} — do powtórzenia r: dokładnie n razy; co najmniej n razy: od n do m razy.

Struktury kontrolne

- if (warunek) instrukcje [else instrukcje]
- while (warunek) instrukcje
- do instrukcje while (warunek)
- for (expr1; expr2; expr3) instrukcje
- for (zmienna in tablica) instrukcje
  - pętla for do iteracji tablic, przykładowo, for (z in T) { obsługa T[z] }

Struktury kontrolne

- break — przerwanie wykonania pętli,
- continue — przerwanie wykonania iteracji petli,
- delete tablica[indeks] — usunięcie elementu tablicy o numerze indeks,
- delete tablica — usunięcie tablicy,
- exit [wyrażenie] — wyjście z kodem równym wyrażeniu,
- { instrukcje } — blok instrukcji.

Predefiniowane zmienne

- $0 – cały przetwarzany wiersz
- $1, $2,... - kolejne pola przetwarzanego wiersza
- NR — liczba przetworzonych rekordów (wierszy),
- FNR — liczba przetworzonych rekordów w bieżącym pliku
- FILENAME — nazwa aktualnie przetwarzanego pliku,

Predefiniowane zmienne

- FS — separator pól w pliku wejściowym,
- OFS — separator pól w pliku wyjściowym,
- RS — separator rekordów w pliku wejściowym,
- ORS — separator rekordów w pliku wyjściowym,
- NF — liczba pól w bieżącym rekordzie,
- ARGC — ilość argumentów skryptu,
- ARGV — tablica argumentów skryptu.

Obsługa wejścia/wyjścia

- next — wstrzymuje przetwarzanie bieżącego i przechodzi do następnego rekordu,
- nextfile — wstrzymuje przetwarzanie bieżącego i przechodzi do następnego pliku,
- print, print lista-wyrazen — drukuje cały wiersz lub drukuje argumenty, np. print $1 $2,
- printf format, lista-wyrazeń — drukuje listę wyrażeń wg ustalonego formatu; instrukcja bardzo podobna do tej z bash-a.

Funkcje matematyczne

- sin(expr), cos(expr), tan(expr)
- log(expr), sqrt(expr)
  - logarytm naturalny, pierwiastek kwadratowy,
- int(expr) — część całkowita z argumentu,
- rand()
  - funkcja zwraca liczbę z przedziału [0*;* 1),
- srand([expr])
  - działa jak funkcja rand(), przy czym można przekazać „ziarno” dla generatora liczb losowych
  - jeśli tego nie zrobimy, to wykorzystywany jest bieżący czas dnia.

Funkcje do obsługi napisów

- gensub(r, s, h [, t]) — funkcja charakteryzująca się następującymi własnościami:
  - zwraca napis, przy czym argument t pozostaje niezmieniony,
  - szuka wyrażenie regularnego r w t wymieniając znalezione na s,
  - argument h określa, które wystąpienie ma być wymienione;
  - jeśli podamy g lub G, to zostaną wymienione wszystkie wystąpienia
  - jeśli nie podamy t, to wykorzystywane jest $0, czyli cały wiersz,

Funkcje do obsługi napisów

- gsub(r, s [, t]) — działa podobnie do funkcji gensub() przy czym zwraca ilość zamian, a zamiany dokonywane są w t (funkcja modyfikuje t),
- sub(r, s [, t]) — działa podobnie do funkcji gsub() przy czym wymienia tylko pierwsze wystąpienie,
- index(s, t) — zwraca miejsce wystąpienia t w s lub 0, jeśli nie występuje (znaki napisów numerujemy od 1),

Funkcje do obsługi napisów

- length([s]) — zwraca długość s lub długość $0, jeśli s nie podamy
- split(s, a [, r]) — rozbija s względem r (jeśli nie podamy, to względem FS) i tworzy z tych kawałków tablice a,
- substr(s, i [, n]) — zwraca s od pozycji i; zwraca co najwyżej n znaków, jeśli pominiemy n zwracany jest s do końca,
- tolower(str), toupper(str) — zwracana jest kopia str z wymienionymi literami na małe (wielkie).

Tworzenie funkcji

- Składnia
  - `function name(lista-parametrów)`
    `{ instrukcje }`
- Przykład
  - ```
    function f(p, q,       a, b)
    # a i b są lokalne
    {
       ...
    }
    ```
  - `{ ... ; f(1, 2) ; ... }`


    ```

# Pierwszy skrypt

- Tworzymy plik z rozszerzeniem .awk
- Plikowi nadajemy prawa do wykonywania.
- W pierwszym wierszu skryptu wpisujemy #!/usr/bin/awk -f.
- Najpierw umieczamy deklaracje funkcji, a potem wzorce z odpowiednimi akcjami.

 

# Przykłady

- Przykład 1

```
#!/usr/bin/awk -f
function silnia(n)
{
  if ((n==0) || (n==1)) return 1;
  else return n*silnia(n-1);
}
{ print silnia($0) }

roadrunner:~ # echo 5 | ./silnia.awk
roadrunner:~ # echo -e "2\n5" | ./silnia.awk
```

- Przykład 2

```
roadrunner:~ # cat dane.txt | awk \
  '/^a.*/  { print $0 }
   /^an.*/ { print $0 }'
```


Przykłady

- Przykład 3

```
roadrunner:~ # ls -l | awk '
BEGIN { s=0 }
/^-/ { s+=$5 }
END { print s }'
```

- Przykład 4

```
roadrunner:~ # cat liczby.txt | awk ' { s=0;
  for (i=1;i<=NF;i++) { s+=$i }; print s; }
```

- Przykład 5

```
roadrunner:~ # echo "ala ma kota i ma psa" |
awk '{ print gensub("ma", "nie ma", "g", $0) }'

ala nie ma kota i nie ma psa
```


Przykłady

- Przykład 6

```
roadrunner:~ # echo "ala ma kota" | awk '
{ split($0,a); print a[2] }'

ma
```

- Przykład 7

```
roadrunner:~ # awk '
!/.*A.*/ && /.*M.*/ { print $1 " " $2 " " $3; }
' dane.txt

2 Magda 120
```

