Politechnika Wrocławska

Systemy operacyjne 2

Laboratorium nr 6

Program `awk`

Szymon Datko

szymon.datko@pwr.edu.pl

Wydział Informatyki i Telekomunikacji,

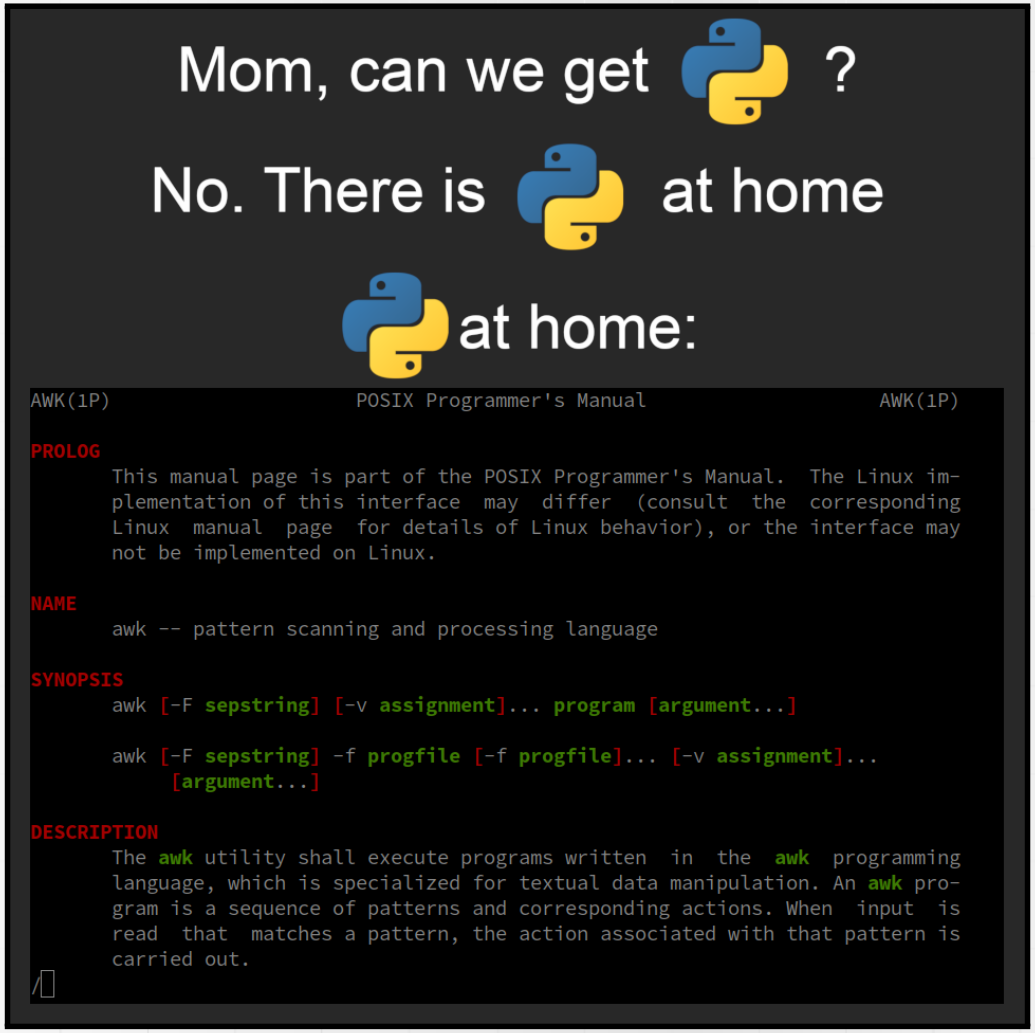
Politechnika Wrocławska

semestr letni 2021/2022

HR EXCELLENCE IN RESEARCH

W skrócie – awk

- Narzędzie do przetwarzania ustrukturyzowanych danych tekstowych.
- W zasadzie to jest to pełnoprawny język programowania.



Źródło obrazka: https://www.reddit.com/r/ProgrammerHumor/comments/d8393s/ (modyfikacja własna)

Politechnika Wrocławska

Część I

Omówienie zagadnień

Program `awk` – informacje ogólne

- Nazwa pochodzi od nazwisk jego autorów: Aho, Weinberger i Kernighan.
- Do działania potrzebny jest interpreter – np. **gawk** (GNU awk).
  - **Uwaga**! Czasem domyślnie może być zainstalowany np. **mawk**.
  - Ta ostatnia implementacja miewa problemy ze złożonymi wyrażeniami.
- Typowo używany jest do przetwarzania danych tekstowych i obliczeń.
- Dane wejściowe dzielone są na tak zwane **rekordy** i **pola**.
- Domyślne separatory:
  - dla rekordów jest to znak nowej linii (`\n`),
  - dla pól są to białe znaki inne niż nowa linia (spacja, tabulator).
    - Domyślnie wielokrotne wystąpienia separatora pól są ściskane.
    - Nie musi tak być, kiedy zdefiniujemy własne separatory.
- Przykładowe uruchomienie:
  - `awk 'instrukcje' plik-do-przetworzenia`
  - `awk -f 'instrukcje.awk' plik-do-przetworzenia`
  - `... | awk 'instrukcje'`
  - Możliwy shebang w pliku z instrukcjami: `#!/usr/bin/env awk -f`

Program `awk` – struktura programów

- Składnię narzędzia `awk` można opisać jako zbiór następujących zapisów:
  - `wzorzec { akcje }`
- Słowem wyjaśnienia:
  - `wzorzec` określa kiedy dany zestaw instrukcji ma zostać wykonany,
  - `akcje` opisują w jaki sposób przetworzyć rekord pasujący do wzorca.
- Na akcje mogą składać się operacje na zmiennych i wywołania.
  - Domyślnie jedna instrukcja to jedna linia programu.
  - Używając średnika `;` można zapisać kilka instrukcji w jednej linii.
    - Podobnie jak robi się to w powłoce **Bash** i języku **Python** ;-)
- Zaawansowane użycie umożliwia definiowanie własnych funkcji, itd.
  - Raczej nie będzie to konieczne na potrzeby zajęć.

Program awk – wzorce

- Wzorce pozwalają kontrolować dla których rekordów wykonać akcje.
  - **Uwaga**! Kilka wzorców może pasować do tego samego rekordu.
  - Zestawy instrukcji są wtedy wykonywane w kolejności w ich deklaracji.
- Brak wzorca = dane akcje wykonane będą dla każdego rekordu.
  - `{ print NR }`
- Specjalny wzorzec **BEGIN** = operacje przed przetworzeniem wejścia
  - `BEGIN { RS = '\n'; FS = ','; SUMA = 0 }`
- Specjalny wzorzec **END** = instrukcje do wykonania na koniec.
  - `END { print SUMA }`.
- Warunki wykorzystujące zmienne i wyrażenia logiczne.
  - `NR % 2 == 0 && length($0) > 5 { print NR, NF }`
- Dopasowanie rekordu / pola lub zmiennej do wyrażenia regularnego.
  - `/wyrażenie szukane w rekordzie/ { print NR }`
  - `$0 ~ /wyrażenie szukane w zmiennej lub polu/ { print NR }`

Program `awk` – akcje

- Akcje służą do jakiegoś przetworzenia pasujących rekordów.
- Jeśli podano same wyrażenie, domyślną akcją jest wyświetlenie rekordu.
  - Czyli: `{ print }`
  - Np. wyświetlenie co drugiego wiersza: `cat plik | awk 'NR % 2'`
- Polecenie `print` służy zwróceniu wartości na standardowe wyjście.
  - Wykonanie zakończone jest separatorem, zdefiniowanym w zmiennej `ORS`.
  - Można podać kilka wartości, rozdzielając je przecinkami.
    - Na wyjściu będą one rozdzielone separatorem pól (zmienna `OFS`).
- Dostępny jest szereg wbudowanych funkcji, pętle i instrukcje warunkowe.
- Akcję może też stanowić obliczenie wartości jakiegoś wyrażenia.
  - Standardowe operatory: `+`, `-`, `!`, `*`, `/`, `^`, `**`, `<`, `>`, `>=`, `<=`, `==`, `!=`, itd.
  - Zazwyczaj w połączeniu z operatorem przypisania `=` do **zmiennej**.

Program `awk` – zmienne

- Są dynamicznie tworzone w razie potrzeby i nie mają stałych typów.
  - Interpretacja wartości zmiennych zależy od kontekstu użycia.
- Nazwę zmiennej może stanowić ciąg znaków z zakresu `[A-Za-z0-9_]`.
  - W odróżnieniu od powłoki **Bash**, nie stosujemy znaku `$` w odwołaniu.
  - Dowolny ciąg poza apostrofami/cudzysłowami to odwołanie do zmiennej.
  - Wartość pod apostrofami/cudzysłowami to typ łańcuchowy (`'Ala ma kota'`).
- Ważne zmienne wbudowane i ich znaczenie:
  - `RS` – przechowuje separator rekordów,
  - `FS` – definiuje separatory pól rekordów,
  - `NR` – zawiera informację o numerze aktualnie przetwarzanego rekordu,
  - `NF` – przechowuje informację o liczbie pól w bieżącym rekordzie.
- Operator `$` pozwala odnieść się do konkretnego **pola** w danym **rekordzie**.
  - Na przykład: `$1`, `$2`, `$NF`, `$MOJEPOLE` (gdy `MOJEPOLE` zawiera liczbę).
  - Przypadek specjalny – cały rekord: `$0`.
  - Do pola można przypisać nową wartość (zmienić ją) i nie jest to błąd!

Program awk – tablice

- Specjalny rodzaj zmiennych, które przechowują pary klucz–wartość.
- Implementacja obejmuje wyłącznie tablice asocjacyjne.
  - Indeksy liczbowe są traktowane jako łańcuchy znaków.
  - Tablice wielowymiarowe są symulowane przez sklejenie indeksów.
    - Stosowany jest tutaj osobny separator – zmienna `SUBSEP`.
- Odwołanie do elementu tablicy realizuje operator `[]`.
  - `tablica[klucz] = 'moja wartość'`
  - `macierz[i, j] = 42`
- Polecenie `delete tablica[klucz]` usuwa element z tablicy.
  - Jeśli poda się samą nazwę tablicy, usunięte zostaną wszystkie elementy.

Program awk – funkcje

- Szereg wbudowanych funkcji tekstowych i matematycznych.
  - Dostępne są także funkcje do manipulacji plikami, czasem, bitami, itd.
  - Oprócz tego użytkownik ma możliwość definiowania własnych funkcji.
- Ważne funkcje wbudowane na potrzeby bieżących zajęć:
  - `length(łańcuch)` – zwraca długość podanego łańcucha.
  - `index(łańcuch, ciąg)` – zwraca pozycję danego ciągu w łańcuchu.
    - Wartość `0` oznacza, że ciągu nie znaleziono w łańcuchu.
  - `split(łańcuch, tablica, separator)` – dzieli łańcuch na części.
    - Wynik zostaje zapisany w zmiennej tablicowej o nazwie `tablica`.
  - `substr(łańcuch, pozycja, długość)` – zwraca fragment łańcucha.
    - Argument `długość` jest opcjonalny – domyślnie to koniec łańcucha.
  - `tolower(łańuch)` – zwraca kopię napisu z zamienionymi literami na małe.
  - `toupper(łańcuch)` – analogicznie jak wyżej, ale zamiana na duże litery.
- Więcej: https://www.gnu.org/software/gawk/manual/html_node/Built_002din.html

Program awk – pętle i instrukcje warunkowe

- Podstawowa składnia jest podobna do znanej z języka **C**.
  - Istotne są nawiasy `()` w zapisie przy `if` i `for`.
  - Instrukcje do wykonania (po `)` i `else`) można przenieść do nowej linii.
  - Przy pomocy klamer `{}` można zagnieździć blok kilku instrukcji.
- Instrukcja warunkowa `if`.
  - `if ( warunek ) instrukcja1; else intrukcja2`
  - Alternatywnie można zapisać `warunek ? akcja1 : akcja2`.
  - Nie ma innych wariantów, ale można zapisać `else if`.
- Pętla `for`.
  - `for ( licznik ; warunek ; akcja ) instrukcja`
  - Np. `for ( i = 1; i <= NF; i++ ) { print $i }`.
- Zapisy związane z tablicami:
  - `if ( klucz in tablica ) instrukcja`
  - `for ( klucz in tablica ) instrukcja`
  - `if ( (i, j) in macierz ) instrukcja`

Interesujący przykład - test kolorów w terminalu

```
[sdatko@polluks ~]$ ./test-terminal-colors.sh
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
[sdatko@polluks ~]$ head -n 16 ./test-terminal-colors.sh
#!/bin/bash
# From: https://gist.github.com/XVilka/8346728

awk 'BEGIN{
    s="/\\/\\/\\/\\/\\"; s=s s s s s s s s;
    for (colnum = 0; colnum<77; colnum++) {
        r = 255-(colnum*255/76);
        g = (colnum*510/76);
        b = (colnum*255/76);
        if (g>255) g = 510-g;
        printf "\033[48;2;%d;%d;%dm", r,g,b;
        printf "\033[38;2;%d;%d;%dm", 255-r,255-g,255-b;
        printf "%s\033[0m", substr(s,colnum+1,1);
    }
    printf "\n";
}'
[sdatko@polluks ~]$
```

Politechnika Wrocławska

Część dla dociekliwych

Zadanie dodatkowe

Politechnika Wrocławska

Zadanie domowe

Tablice w Bashu – krótkie omówienie.

Język Bash pozwala na przechowywanie prostych struktur w zmiennych, takich jak tablice. Proszę omówić dwie podstawowe takie struktury – tablice zwykłe i asocjacyjne: jak się je tworzy, używa i czym one się różnią.

W szczególności proszę opisać testowanie zawartości takich tablic (liczbę elementów), jak iterować po ich zawartości oraz jak dodawać i usuwać elementy. Czym różni się wywołanie `"${tablica[@]}"` od `"${tablica[*]}"`?