[ARCHIWUM] Obsługa awk

> **Strona zarchiwizowana.** Treść została rozbudowana i przeniesiona. Aktualna wersja: awk — język do danych w kolumnach. Ta strona zostaje wyłącznie dla historii.

> AWK to potężne narzędzie do przetwarzania i analizy tekstu, szczególnie przydatne do pracy z danymi podzielonymi na kolumny. Pozwala na wyszukiwanie, filtrowanie, formatowanie i agregację danych z plików tekstowych lub wyników innych poleceń.

Podstawowa składnia AWK

```
awk 'warunek { działania }' plik
```

Wyjaśnienie:

- `warunek` – wyrażenie logiczne lub wzorzec tekstowy, który musi być spełniony, aby wykonać `działania` (np. filtrowanie linii).
- `{ działania }` – polecenia wykonywane na liniach spełniających warunek (np. wypisanie wybranych kolumn).

Odwołania do pól i zmienne w AWK

```
$0       # cała linia tekstu wejściowego
$1, $2, … # kolejne kolumny/ pola w linii (domyślnie pola rozdzielone spacjami)
NR       # numer bieżącego wiersza (count)
NF       # liczba pól w bieżącej linii
```

Wyjaśnienia:

AWK dzieli każdą linię na pola (domyślnie na spacje/tabulacje). Użycie `$1` zwraca pierwszą kolumnę, `$2` drugą itd. Zmienna `NR` oznacza numer bieżącej linii, a `NF` ilość kolumn w bieżącym wierszu.

Przykłady użycia AWK – pojedyncze komendy

1. Wyświetl pierwszą i trzecią kolumnę każdego wiersza:

```
awk '{ print $1, $3 }' plik.txt
```

2. Wyświetl linie, których pierwsza kolumna równa jest "error":

```
awk '$1 == "error" { print $0 }' plik.txt
```

3. Sumowanie wartości w kolumnie numer 2:

```
awk '{ suma += $2 } END { print suma }' plik.txt
```

Wyjaśnienie: dla każdej linii dodaj wartość pola 2 do zmiennej `suma`, a na końcu `END` wypisz wynik.

4. Wypisz numer linii oraz zawartość linii:

```
awk '{ print NR, $0 }' plik.txt
```

Wyrażenia regularne (regexp) w AWK

AWK wspiera wyrażenia regularne do dopasowywania tekstu. Można je stosować w warunkach:

```
awk '/pattern/ { print $0 }' plik.txt
```

Znajduje i wypisuje linie zawierające `pattern`.

Można łączyć takie dopasowanie z innymi warunkami i operatorami:

```
awk '$3 ~ /error/' plik.txt
```

Wypisuje linie, w których trzecia kolumna zawiera słowo "error".

```
awk '$2 !~ /^[0-9]+$/' plik.txt
```

Wypisuje linie, gdzie druga kolumna nie składa się wyłącznie z cyfr.

Przetwarzanie i filtrowanie danych z AWK – bardziej rozbudowane przykłady

Wyświetlanie wierszy, gdzie kolumna 5 jest większa niż 100:

```
awk '$5 > 100 { print $0 }' plik.txt
```

Formatowane wypisywanie danych (np. kolumny 1 i 3 oddzielone tabulatorem):

```
awk '{ printf "%s\t%s\n", $1, $3 }' plik.txt
```

Podliczanie wystąpień unikalnych wartości w kolumnie 3:

```
awk '{ counts[$3]++ } END { for (val in counts) print val, counts[val] }' plik.txt
```

Wywołania specjalne i kontrola przepływu

AWK pozwala na tworzenie bloków `BEGIN` i `END` do wykonania instrukcji przed lub po przeczytaniu wszystkich linii:

```
awk 'BEGIN { print "Start przetwarzania" }
     { print $1 }
     END { print "Koniec przetwarzania" }' plik.txt
```

Pętla i warunki są typowe dla języków programowania, np.:

```
awk '{
  if ($2 > 100)
```

```
    print $0
  else
    next
}' plik.txt
```

Revision #3
Created 2025-11-23 13:34:47 UTC by Tomasz Konieczny
Updated 2026-09-17 20:19:02 UTC by Tomasz Konieczny