

**Politechnika Łódzka**  
**Wydział Elektrotechniki Elektroniki Informatyki i Automatyki**  
**Instytut Informatyki Stosowanej**

**Laboratorium z przedmiotu**  
**Systemy Operacyjne 1**

**Moduł Linux: ćwiczenie nr 6**

Edytory strumieniowe sed i awk

## Spis treści

|                                                                                                                          |    |
|--------------------------------------------------------------------------------------------------------------------------|----|
| 1. Edytor strumieniowy sed.....                                                                                          | 3  |
| 1.1. Obsługa edytora sed .....                                                                                           | 3  |
| 1.1.1. Składnia sed .....                                                                                                | 3  |
| 1.1.2. Skrypty sed.....                                                                                                  | 3  |
| 1.1.3. Polecenia sed .....                                                                                               | 4  |
| 1.1.4. Zadania.....                                                                                                      | 5  |
| 1.2. Wyrażenia regularne w sed .....                                                                                     | 5  |
| 1.2.1. Dopasowywanie .....                                                                                               | 5  |
| 1.2.2. Podstawianie.....                                                                                                 | 6  |
| 1.2.3. Zadania.....                                                                                                      | 7  |
| Uwaga! W poniższych zadaniach proszę nie modyfikować zawartości plików – wyniki zmian należy wyświetlić na konsoli. .... | 7  |
| 1.3. Polecenia edycyjne edytora sed.....                                                                                 | 7  |
| 1.3.1. Spis poleceń .....                                                                                                | 7  |
| 1.3.2. Wyświetlanie i usuwanie.....                                                                                      | 8  |
| 1.3.3. Podstawianie .....                                                                                                | 9  |
| 1.3.4. Odczyt i zapis do pliku .....                                                                                     | 10 |
| 1.3.5. Zadania.....                                                                                                      | 10 |
| 2. Język awk .....                                                                                                       | 10 |
| 2.1. Obsługa interpretera awk, wykonywanie skryptów.....                                                                 | 10 |
| 2.1.1. Składnia poleceń.....                                                                                             | 11 |
| 2.2. Zmienne w awk .....                                                                                                 | 12 |
| 2.3. Interakcja z użytkownikiem w awk .....                                                                              | 13 |
| 2.4. Operacje logiczne i arytmetyczne w awk .....                                                                        | 13 |
| 2.4.1. Operacje logiczne .....                                                                                           | 13 |
| 2.4.2. Operacje arytmetyczne.....                                                                                        | 14 |
| 2.5. Wybrane struktury języka awk.....                                                                                   | 14 |
| 2.5.1. Pętla while.....                                                                                                  | 14 |
| 2.5.2. Pętla do/while .....                                                                                              | 15 |
| 2.5.3. Pętla for.....                                                                                                    | 15 |
| 2.6. Zadania .....                                                                                                       | 16 |

# 1. Edytor strumieniowy sed

## 1.1. Obsługa edytora sed

### 1.1.1. Składnia sed

Edytor sed jest to nieinteraktywny edytor używany w skryptach do filtrowania informacji w pliku. Składnia poleceń została zaczerpnięta z edytora liniowego *ed*, dostępnego w większości systemów unixowych. Sed, w przeciwieństwie do innych edytorów, nie modyfikuje pliku, lecz wysyła wynik operacji na standardowe wyjście.

Składnia edytora sed:

```
sed -n 'polecenie edycji' [plik_wejściowy]
sed -n [-e polecenie edycji] [-f skrypt_z_poleceniami] [plik_wejściowy]
```

gdzie opcje mają następujące znaczenia:

- n  
nie wyświetlaj wynikowych wierszy (brak tej opcji spowoduje wyświetlenie wszystkich wierszy !)
- e  
pobierz polecenie z wiersza poleceń. Parametr nie jest wymagany, jeśli ma zostać wykonane jedno polecenie. Jeśli chcemy wykonać kilka poleceń, każde z nich musi być poprzedzone '-e'.
- f  
pobierz polecenia z pliku

Przykłady równoznacznych wywołań:

```
sed -e 's/ala/Ala/' dane >dane2
sed 's/ala/Ala/' dane >dane2
sed 's/ala/Ala/' <dane >dane2
cat dane | sed 's/ala/Ala/' > dane2
```

w przykładzie zastosowano najczęściej używane polecenie podstawienia (s).

### 1.1.2. Skrypty sed

Aby wykonać kilka poleceń na tych samych danych, najlepiej zapisać odpowiednie komendy do pliku tekstowego. W każdej linii powinna znaleźć się oddzielna komenda. Uruchomienie takiego pliku możliwe jest poprzez wydanie polecenia:

```
sed -f skrypt plik_wejściowy > plik_wyjściowy
```

Bardzo pomocne może być dołożenie w pierwszej linii skryptu wpisu: `#!/bin/sed -f`. Pozwoli on na uruchamianie skryptu jak pliku wykonywalnego. Skrypt taki może mieć postać:

```
#!/bin/sed -f  
s/a/A/g s/e/E/g  
s/i/I/g s/o/O/g  
s/u/U/g
```

Następnie plikowi ze skryptem można nadać prawo wykonania:

```
chmod a+x skrypt
```

i uruchomić:

```
./skrypt
```

### 1.1.3. Polecenia sed

Polecenie sed ma następującą składnię:

```
[zakres_wierszy] [polecenie_edycyjne] [argumenty_polecenia]
```

#### 1.1.3.1. zakresy wierszy

Zakres wierszy opisuje wiersz, od którego ma być rozpoczęte przetwarzanie i wiersz na którym ma się zakończyć. Zakres wierszy można podać określając numer pierwszego i ostatniego wiersza, rozdzielając je przecinkiem:

```
1,5
```

Można także określić go, podając tekst, który ma zawierać pierwszy i ostatni wiersz:

```
/ala/, /kota/
```

Ewentualnie, jeśli nie jesteśmy w stanie dokładnie określić tekstu, możemy posłużyć się wyrażeniem regularnym `/wyr_pocatkowe/`, `/wyr_koncowe/`. Na przykład:

```
/abc[0-9]/, /a.*z/
```

W przypadku jawnych tekstów i wyrażeń regularnych zakresy powinny być ujęte w znaki "/".

Przykład zastosowania zakresu:

```
sed '5,20 s/ala/Ala/' dane >dane2
```

### 1.1.3.2. Polecenia edycyjne

Polecenia edycyjne stanowią pojedyncze litery oraz następujące po nich listy argumentów. Argumenty powinny być oddzielone od polecenia oraz od siebie separatorem. Najczęściej spotykanym separatorem jest znak "/". Ponieważ znak "/" ma wówczas specjalne znaczenie każde inne wystąpienie "/" powinno być poprzedzone znakiem "\". Ewentualnie można zastosować inny separator. Poniższe przykłady wykonają tę sama akcję:

```
sed 's/\\usr\\local\\bin\\/common\\bin/' <old >new  
  
sed 's_/usr/local/bin_/common/bin_' <old >new  
  
sed 's:/usr/local/bin:/common/bin:' <old >new  
sed 's|usr/local/bin|common/bin|' <old >new
```

Inne polecenia zostaną omówione w dalszej części instrukcji.

### 1.1.4. Zadania

1. Zapoznaj się z dokumentacją edytora sed
2. Spróbuj utworzyć przy pomocy edytora sed dowolny plik tekstowy
3. Sprawdź działanie komendy podstawienia "s"
4. Sprawdź różnice w działaniu pomiędzy wywołaniem kilku poleceń przy pomocy przełączników -e, wykonanych w skrypcie oraz po uruchomieniu wielu instancji seda:

```
cat dane | sed [...] | sed [...] | sed [...] > dane2
```

## 1.2. Wyrażenia regularne w sed

### 1.2.1. Dopasowywanie

Wyrażenie regularne w poleceniu sed może wystąpić zarówno w wyróżniku zakresu, jak i argumentcie. Wyrażenie regularne seda są rozwinięciem wyrażeń udostępnianych przez komendę grep:

| Znak  | Opis                                                                                                                                   |
|-------|----------------------------------------------------------------------------------------------------------------------------------------|
| .     | Dopasuj dowolny znak                                                                                                                   |
| \$    | Dopasuj poprzedzające wyrażenie do końca wiersza                                                                                       |
| ^     | Dopasuj występujące po operatorze wyrażenie do początku wiersza                                                                        |
| *     | Dopasuj zero lub więcej wystąpień znaku poprzedzającego operator                                                                       |
| \     | Oznacza pominięcie specjalnego znaczenia znaku np.: \*                                                                                 |
| [ ]   | Dopasuj dowolny znak ujęty w nawiasy. np.: [abc]                                                                                       |
| [ - ] | Dopasuj dowolny znak z przedziału. np.: [0-9] – wszystkie cyfry; [a-z] – wszystkie małe litery; [0-9a-zA-Z] – wszystkie litery i cyfry |
| [ ^ ] | Dopasuj znak, który nie znajduje się w nawiasach. np.: [^abc] lub [^ ] - dowolny znak nie będący spacją                                |

Przykłady:

```
grep 'Ala' plik      #znajduje wyraz Ala
grep 'A.a' plik      #znajduje wyrazy takie jak Ala, Aga, Ara, A+a i inne
grep 'A[lg]a' plik   #znajduje TYLKO wyrazy Ala i Aga
grep '^Ala' plik     #znajduje linię 'Ala ma kota.' ale odrzuca 'To jest Ala.'
grep 'Go*gle' plik   #znajduje Gogle, Google, Gooogle itd.
grep '[0-9][0-9]*'   #znajduje dowolny ciąg cyfr
```

### 1.2.2. Podstawianie

W przypadku kiedy wyrażenie występuje w argumencie, można zastosować bufory w których zostaną zapamiętane odszukane ciągi. Ciąg zapamiętany w pierwszym argumencie komendy (n.p. podstawienia) można wstawić w drugim. Aby zapamiętać dopasowany tekst, należy odpowiadające mu wyrażenie regularne ująć w znaki "(" oraz ")". Aby wstawić zapamiętany ciąg należy użyć wyrażenia "\1". Na przykład:

```
sed 's/(Ala ma\) [^ ]*/\1 psa/'
```

Wydanie polecenia:

```
echo Ala ma kota i papuge | sed 's/(Ala ma\) [^ ]*/\1 psa/'
```

zwróci:

```
Ala ma psa i papuge
```

### 1.2.3. Zadania

**Uwaga! W poniższych zadaniach proszę nie modyfikować zawartości plików – wyniki zmian należy wyświetlić na konsoli.**

1. W pliku /etc/group zamień słowo root na admin
2. W pliku /etc/group zamień wszystkie cyfry/liczby na ciąg 'xxxx'
3. W pliku /etc/group zamień grupę root na admins - nazwy grup występują jedynie na początku wiersza, dalsze wpisy oznaczają użytkowników
4. W pliku /etc/passwd wyszukaj ścieżki bezwzględne (zaczynające się od "/") i zastąp je ciągiem "-----". Czy wszystkie ścieżki w linii zostały zastąpione?
5. W pliku /etc/passwd wyszukaj ścieżki bezwzględne (zaczynające się od "/") umieść je w nawiasach "[" oraz "]" (należy skorzystać z podstawiania)

## 1.3. Polecenia edycyjne edytora sed

### 1.3.1. Spis poleceń

Polecenia programu sed zostały zaczerpnięte z edytora liniowego ed. Ponieważ pełniły one funkcję skrótów klawiszowych, polecenia oznaczane są pojedynczymi literami. Poniżej przedstawiono listę poleceń sed wraz z krótkim opisem

|            |                                                  |
|------------|--------------------------------------------------|
| a tekst    | dopisuje tekst na początku wiersza               |
| b etykieta | przejdź do polecenia zaczynającego się :etykieta |
| c tekst    | zastępuje wiersze tekstem                        |
| d          | usuwa wiersze                                    |
| p          | wyświetla wiersz                                 |
| g          | zastępuje bieżący wiersz zawartością bufora      |
| h          | kopiuje wiersz do bufora                         |
| i tekst    | wstawia tekst przed wybranymi wierszami          |
| r nazwa    | wysyła na wyjście plik o danej nazwie            |

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| s/wyr_reg/tekst/znaczniki | <p>zastępuje ciąg odpowiadający wyrażeniu regularnemu na tekst. Znaczniki:</p> <ul style="list-style-type: none"> <li>▪ brak - zastąp pierwsze wystąpienie wzorca w linii)</li> <li>▪ g – zastąp wszystkie wystąpienia wzorca;</li> <li>▪ p – wyświetl wiersz po wykonaniu zastąpienia</li> <li>▪ w – zapisz wiersz do pliku po wykonaniu zastąpienia</li> <li>▪ n – zastąp n-te wystąpienie wzorca</li> </ul> <p>symbol &amp; oznacza ciąg pasujący do wyrażenia regularnego</p> |
| t etykieta                | przejdź do wiersza z etykietą po wykonaniu zastąpienia                                                                                                                                                                                                                                                                                                                                                                                                                            |
| w nazwa                   | zapisz bieżący wiersz do pliku                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| y /abc/xyz                | zastąp odpowiednie znaki (porównaj polecenie tr)                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| =                         | wypisz numer wiersza na wyjściu                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| !polecenie                | wykonaj polecenie gdy wiersz nie odpowiada wzorcowi                                                                                                                                                                                                                                                                                                                                                                                                                               |
| :etykieta                 | etykieta                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| { }                       | grupa poleceń                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

Dodatkowo każda komenda może zostać poprzedzona znakiem "!". Oznacza to, iż dotyczy linii nie pasujących do podanych zakresów. Na przykład:

```
ps | sed -n '/bash/ !p'
```

Zakres określa linie zawierające słowo "bash". Na ekranie wyświetlą się jedynie linie (w tym przypadku procesy), które tego słowa nie zawierają.

### 1.3.2. Wyświetlanie i usuwanie

Uruchomienie programu sed z opcją -n pozwala na kontrolowanie, które linie mają zostać wyświetlone, a które nie. Aby określona linia została wysłana na wyjście, można skorzystać z komendy "p". Do wyboru linii najlepiej posłużyć się zakresami. Na przykład, aby wyświetlić linie od 10 do 15 napiszemy:

```
sed -n '10,15 p'
```

Usuwanie "d" ma podobne zastosowanie do wyświetlania "p". Jednakże ich działanie jest przeciwstawne. Aby uzyskać identyczne działanie jak w poprzednim przykładzie, należy zastosować negację:

```
sed '10,15 !d'
```



Proszę zwrócić uwagę, iż w tym przypadku nie należy stosować parametru -n.

### 1.3.3. Podstawianie

Polecenie podstawienia składa się z 4 części:

1. komenda - polecenie s
2. szukany wzorzec - wyrażenie regularne, którego wystąpienia szukamy. Domyślnie dopasowywane jest tylko pierwsze wystąpienie ciągu w każdej linii. Dodatkowo, oprócz standardowych operatorów wyrażeń regularnych, można stosować znaczniki "(" oraz ")".
3. podstawiany wzorzec - tekst wstawiany w miejsce odszukanego. Można stosować znaczniki "/", "2", .... podstawiające teksty dopasowane do kolejnych wyrażenia w nawiasach "( )". Jeśli w ciągu odszukanym nie użyto powyższych nawiasów, można podstawić cały ciąg wyszukany znacznikiem "&". Przykład:

```
echo 'Ala Ania Ola Ela Zosia' | sed 's/.la/<&>/g'
```

zwróci:

```
<Ala> Ania <Ola> <Ela> Zosia
```

#### 4. znaczniki -

- g – zastąp wszystkie wystąpienia wzorca w linii; Jeśli we wcześniejszym przykładzie pominęto by "g", efekt byłby następujący:

```
<Ala> Ania Ola Ela Zosia
```

- p – wyświetl wiersz po wykonaniu zastąpienia. Ma zastosowanie w przypadku użycia opcji -n przy uruchomieniu seda
- w – zapisz wiersz do pliku po wykonaniu zastąpienia. Przykład:

```
echo 'Ala Ania Ola Ela Zosia' | sed -n 's/.la/<&>/gw wynik.txt'
```

Wynik działania trafi do pliku wynik.txt

- n – zastąp n-te wystąpienie wzorca. Przykład:

```
'Ala Ania Ola Ela Zosia' | sed 's/.la/<&>/3'
```

zwróci

```
Ala Ania Ola <Ela> Zosia
```

### 1.3.4. Odczyt i zapis do pliku

Podczas przetwarzania pliku przez sed możliwe jest zapisanie wybranych linii do pliku lub wstawienie danych z pliku zewnętrznego. Aby wstawić dane z zewnętrznego pliku można posłużyć się komendą "r". Przykład:

Utwórzmy plik:

```
echo '^^^^^^ do usunięcia' > dane.txt
```

a następnie użyjmy go:

```
cat /etc/group | sed '/root/ r dane.txt'
```

Polecenie 'w' pozwala na zapis wybranych linii do pliku. Umożliwia to na przykład podział pliku według zadanego kryterium:

```
cat /etc/group | sed -n '/root/ w grupy_roota.txt'
```

### 1.3.5. Zadania

1. Z plików /usr/share/X11/xorg.conf.d/\*.conf wyświetl sekcje rozpoczynające się od Section "OutputClass" a kończące EndSection
2. Z pliku /etc/hosts wyświetl linie nie zaczynające się od znaków "#" i ":"
3. Wyświetl listę plików i katalogów z katalogu /etc, które zakończone są .d. Zastąp .d ciągiem .config.
4. Napisz skrypt, który z plików /usr/share/X11/xorg.conf.d/\*.conf zapisze sekcje "OutputClass" do pliku outputs.txt, a sekcje "InputClass" do pliku inputs.txt.

## 2. Język awk

### 2.1. Obsługa interpretera awk, wykonywanie skryptów

Awk jest językiem ogólnego przeznaczenia do przeszukiwania wzorców i przetwarzania tekstów. Można za jego pomocą wykonywać różnorodne działania związane z filtrowaniem, przekształcaniem i sporządzaniem raportów.

Interpreter awk ma następującą składnię:

```
awk [-Fseparator_pól] 'program' nazwa_pliku  
awk [-Fseparator_pól] -f plik_z_programem nazwa_pliku
```

Pierwsza wersja pozwala na wykonanie prostych skryptów awk bezpośrednio z linii poleceń powłoki. W drugim przypadku wykonywany jest skrypt zapisany w pliku. Istnieje też możliwość wykorzystania własności powłoki systemowej, pozwalającej na automatyczne wykonanie skryptu. W tym celu należy w pierwszym wierszu skryptu wpisać:

```
#!/usr/bin/awk -f
```

na początku skryptu i ustawić plikowi prawo wykonania. Możemy wówczas posługiwać się skryptem jak normalnym plikiem wykonywalnym:

```
./nazwa_skryptu nazwa_pliku_do_przetwarzania
```

### 2.1.1. Składnia poleceń

Program awk składa się z listy poleceń oddzielonych średnikiem lub końcem linii:

```
polecenie1  
polecenie2  
polecenie3
```

lub

```
polecenie1; polecenie2; polecenie3
```

Każde polecenie awk przetwarzane jest dla każdej linii pliku wejściowego. Format pojedynczego polecenia awk ma postać:

```
[warunek] [{procedura} ...]
```

Zarówno warunek jak i procedura są opcjonalne.

Istnieją dwa specjalne warunki: BEGIN i END. BEGIN jest prawdziwe przed rozpoczęciem przetwarzania pliku, zaś END po zakończeniu. Pozwalają one na wykonanie polecenia rozpoczynającego i kończącego przetwarzanie pliku. Przykład – poniższe polecenie wyświetli plik, rozpoczynając go tekstem „początek” i kończąc tekstem „koniec”:

```
awk 'BEGIN {print "początek"}; {print $0} ; END {print "koniec"}' nazwa_pliku
```

lub skrypt:

```
BEGIN {print "początek"}
{print $0}
END {print "koniec"}
```

W dalszej części przykłady wywołania prostych poleceń będą pokazywane w postaci skryptów.

## 2.2. Zmienne w awk

Zmienne w awk wypełniane są automatycznie podczas odczytu danych. Wczytany plik automatycznie dzielony jest na pola, na podstawie podanego znaku oddzielającego. Do każdego z pól przypisywany jest identyfikator \$1 do \$n. Identyfikator \$0 oznacza cały wiersz. Przykład:

```
awk -F: '{print $1, $3}' plik
```

Parametr *-F* określa separator na podstawie którego dokonywany jest podział. Domyślnie są to spacje. W kodzie można zmienić separator pól modyfikując zmienną 'FS'. Podobną do FS funkcję pełni zmienna RS - określa ona separator kolejnych linii (rekordów). Przykłady modyfikacji tych zmiennych:

```
FS="\t+"          #tabulator występujący 1 lub więcej razy (+)
FS=":"           #pojedynczy dwukropek
FS="[[:space:]]+" #spacja występująca 1 lub więcej razy
FS="a[ldg]a"      #wyrażenie regularne

RS="\n"
RS=";"
```

Inną często używaną zmienną wbudowaną jest *NR*. Określa ona numer aktualnie przetwarzanej linii. Oto przykład zastosowania:

```
{ print NR, "> ", $0 }
```

W awk istnieje także możliwość definiowania własnych zmiennych. Nie wymagają one deklaracji, a ich typ określany jest w chwili przypisania, np.:

```
{x = 3}
```

Dostępne są także zmienne tablicowe (indeksowane od 1):

```
{
  zm[1] = 10
  zm[2] = 20
  print zm[2]
}
```

## 2.3. Interakcja z użytkownikiem w awk

W przypadku awk, interakcja z użytkownikiem jest dość ograniczona. Wynika to z faktu, iż język przewidziany jest do obróbki wsadowej danych z plików. Do wyświetlania informacji służy polecenie 'print'. Na przykład:

```
BEGIN{print "hello world"}
```

Dane do przetwarzania pobierane są ze standardowego wejścia i kolejne linie przetwarzane są przez każdą instrukcję skryptu. Mogą być zatem także pobierane bezpośrednio od użytkownika (działanie programu kończymy Ctrl-D):

```
awk '{print "Wpisano: ", $0}'
```

## 2.4. Operacje logiczne i arytmetyczne w awk

### 2.4.1. Operacje logiczne

Operacja logiczna "wbudowana" jest w każdą instrukcję. Na jej podstawie podejmowana jest decyzja, czy dana instrukcja ma przetworzyć bieżącą linię tekstu. Ponieważ awk ma charakter filtra, jednym z rodzajów warunków jest wyrażenie regularne. Ma on wówczas postać:

```
/wyrażenie_regulane/ procedura
```

Wyrażenia regularne awk są rozwinięciem wyrażeń regularnych programu grep. Wyrażenie regularne oznacza się ujmując je w znaki "/". Przykład – wyszukiwanie określonych wzorców:

```
/USA/ {print $0}  
/^TOTAL/
```

Brak procedury, jak w drugim przykładzie, spowoduje wyświetlenie całej linii. Jest zatem równoważny z procedurą *print \$0*.

Kolejnym typem wzorców są wyrażenia relacji. Wynik jest drukowany, jeżeli spełniony jest warunek relacji. Możliwymi operatorami są: *=*, *<*, *<=*, *>*, *>=*, *!=* oraz *~* i *!~* który zwraca prawdę jeżeli lewy operand zawiera (nie zawiera) wyrażenie regularne zapisane w drugim operandzie. Przykład – zwraca wiersze w których pierwszym polem jest „Chapter1”:

```
$1 == "Chapter1"
```

Istnieje możliwość wpisanie wielu wyrażeń lub zakresów połączonych operatorami logicznymi:

wzorzec && wzorzec

iloczyn logiczny ("i", "and")

wzorzec || wzorzec

suma logiczna ("lub", "or")

wzorzec, wzorzec

operator zakresu - prawdziwy dla wszystkich linii znajdujących się pomiędzy dwoma wyrażeniami (lub numerami linii)

! wzorzec

operator NOT

Dla przykładu, poniższy skrypt drukuje zawartość linii nie zaczynających się od „S” i zawierających w drugim polu liczbę większą od 100:

```
{ $0 !~ /^S/ } && { $2 > 100 } { print $0 }
```

## 2.4.2. Operacje arytmetyczne

Awk wykonuje standardowe operacje arytmetyczne:

+ : dodawanie

- : odejmowanie

\* : mnożenie

/ : dzielenie

^ : potęgowanie

% : modulo (reszta z dzielenia)

Ponadto dostępna jest większość operatorów znanych z języka C (+, -, ++, --, ...)

Przykład zastosowania (w pierwszej kolumnie pliku wejściowego powinny znaleźć się liczby):

```
BEGIN { a = 0 }
{
    a = a + $1
    print "Dodaje liczbę ", $1
}
END { print "Suma: ", a }
```

## 2.5. Wybrane struktury języka awk

### 2.5.1. Pętla while

Najprostszą pętlą w awk jest pętla *while*. Wykonuje się ona póki spełniony jest warunek sterujący pętlą. Ma ona następującą składnię:

```
while ( warunek ) {
    blok kodu
}
```

Przykład:

```
{
count=10
while (count != 0) {
    print count
    count = count - 1
}
}
```

### 2.5.2. Pętla do/while

W awk dostępna jest także pętla "do...while", która sprawdza swój warunek na końcu bloku, a nie na początku. Jest podobna do dostępnej w niektórych innych językach pętli "repeat...until". Na przykład:

```
{
count=1
do {
    print "Zostanę wydrukowana co najmniej raz niezależnie od wszystkiego"
} while ( count != 1 )
}
```

(Przykładowy kod zostanie wykonany dla każdej linii w pliku wejściowym. Aby wykonał się tylko raz, należy go poprzedzić słowem *BEGIN*)

Ponieważ warunek jest sprawdzany po wykonaniu bloku, pętla "do...while" zawsze zostanie wykonana co najmniej raz. W przeciwieństwie do tego, zwykła pętla while nigdy nie wykona swojego kodu jeżeli na samym początku warunek nie jest spełniony.

### 2.5.3. Pętla for

Pierwszy wariant pętli *for* wykorzystuje tablice. Na przykład:

```
for ( x in myarray ) {
    print myarray[x]
}
```

Drugi wariant pozwala na tworzenie pętli *for*, które są identyczne jak ich odpowiedniki w C:

```
for ( wstępne przypisanie; warunek; modyfikacja ) {
    blok kodu
}
```

Prosty przykład:

```
{
    for ( x = 1; x <= 4; x++ ) {
        print "iteracja",x
    }
}
```

Powyższy kod wyświetli następujący tekst:

```
iteracja 1
iteracja 2
iteracja 3
iteracja 4
```

## 2.6. Zadania

1. Z pliku `/etc/passwd` wyświetl tylko nazwy użytkowników, ich katalogi domowe oraz powłokę
2. Do danych z poprzedniego punktu dodaj nagłówki "Użytkownik", "Katalog" oraz "Powłoka"
3. Do danych z poprzedniego punktu dodaj numery wierszy (i odpowiedni nagłówek).
4. Do danych z poprzedniego punktu dodaj ramki (posłuż się znakami "-" "|" oraz znakiem tabulacji)
5. Napisz skrypt rysujący wykres słupkowy dla pliku z liczbami. Na przykład dla pliku:

```
4
2
8
```

wykres ma mieć postać:

```
| ****
| **
| *****
```