初心者用
awk 講座

第 2 版 2000 年 2 月 25 日

文責：斎藤輪太郎

1.はじめに

awk はテキスト処理言語です。与えられたテキストファイル(cat や more で中味を見ることができるファイル)を加工して出力する時に大変便利です。このテキストではいくつかの例題をこなしながら awk をある程度使えるようになることを目指します。

2.行単位の処理の基本

以下のようなファイルを作ってみましょう。ファイル名を food.txt とします。

```
I like an apple.
He ate a banana.
I cooked some corn.
She has some donuts
```

次に以下のような awk スクリプトを書いてみましょう。ファイル名を eat.awk とします。

```
/apple/ { print $0"…Ringo!!" }
/corn/  { print $0"…Toumorokoshi!!!" }
```

そして以下のようにして awk を起動します。

```
awk –f eat.awk food.txt
```

この場合、food.txt に対して eat.awk の処理が行われます。以下のような出力が得られましたか？

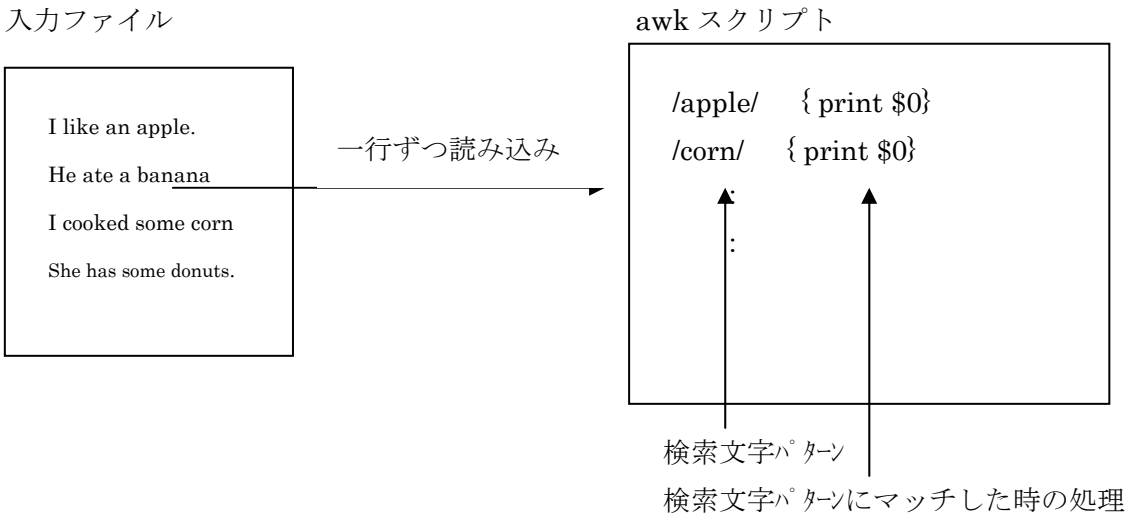
```
I like an apple....Ringo!!!
I cooked some corn....Toumorokoshi!!!
```

awk は一般的に次のような形式をしています。

/検索対象の文字列 1/{ 文字列 1 があった行に対する処理 }
/検索対象の文字列 2/{ 文字列 2 があった行に対する処理 }
:
/検索対象の文字列 n/{ 文字列 n があった行に対する処理 }

そして次のような処理が行われます。

1.入力ファイル（例では food.txt）が一行読み込まれます。
2.その一行の中に文字列 1～n がないか調べられます。もし文字列が見つかればそれに対応する処理が行われます。
3.次の一行を読み込んで同じ処理を繰り返します(1 に戻る)。



では先ほどのスクリプトをみてみましょう。

```
/apple/ { print $0"…Ringo!!" }
/corn/  { print $0"…Toumorokoshi!!!" }
```

1 行目は、読み込んだ行に apple という文字列が含まれていたら、その行($0 で表す)と"….Ringo!!"を表示させる文です。
2 行目は、読み込んだ行に corn という文字列が含まれていたら、その行($0 で表す)と"….Toumorokoshi!!!"を表示させる文です。

課題：スクリプトファイルを以下のように変えて実行してみましょう。

```
/apple/ { print $4"…Ringo!!" }
/corn/  { print $4"…Toumorokoshi!!!" }
```

$4 は処理中の行の 4 番目の単語を表します。

課題：スクリプトファイルを以下のように変えて実行してみましょう。

```
/some/ { print "Ikutsukano "$4}
```

3.BEGIN と END

スクリプトを実行する前の初期設定を BEGIN の中に、スクリプトが終了したときの処理を END の中に書く事ができます。以下のスクリプトを実行してみましょう。入力ファイルはなんでもかまいません。

```
BEGIN { line_counter = 0 }
{ line_counter ++ }
END { print line_counter }
```

入力ファイルの行数が表示されましたか？
Awk ではC言語などと同様に変数が使えます（C言語の int line_counter;のようにあらかじめ宣言する必要はありません。）
１行目で行数を数える変数 line_counter を 0 にセットします。
２行目で行数を一行ずつカウントしていきます(++は line_counter の値を１つ増やすという意味です)。この例では /検索対象の文字列/ が省略されています。これを省略すると全ての行が処理の対象となります。
３行目で最後に行数を出力します。

課題：てきとうに英文のテキストファイルを作りましょう（少し長めに）。次に the という単語が含まれる行数をカウントするような awk スクリプトを作成しましょう。

4.コマンドラインからの引数の受け渡し

awk ではコマンドライン(awk 起動時)に 変数=値 の形があると、この変数が awk のスクリプトに渡されます。以下のような一行のスクリプトを書いてみましょう。このファイル名を p1.awk とします。

```
{ print param1" "$0 }
```

次にこれを以下のように実行してみましょう。

```
awk –f p1.awk param1=">>" 入力ファイル名
```

全ての入力ファイルの行の先頭に >> が付けられるのが分かると思います。上の例では、コマンドラインの param1=">>" で param1 という変数に>>が代入され、print param1" "$0 で全ての入力行の前に>>が（１つスペースを置いて）付加されるようになっています。

課題：上の例を参考に、コマンドラインである文字列を受け取り、全ての入力行の末尾にその文字列を付加するようなスクリプトを書きましょう。

5.正規表現

awk で検索対象の文字列を指定するときには正規表現を使うことができます。正規表現は簡単にかみくだいて言えば、１つの表現で複数のパターンを指定できる表現法です（この説明は情報科学的に正しいとはいえませんが)。
例えば正規表現でma.eはmake,made,maleなど複数のパターンを表します。正規表現を使うことによってパターンの検索に融通が利くようになるわけです。ピリオド"."は任意の一文字を表します。
正規表現の例をいくつか挙げておきます。

^	行頭
.	任意の１文字
[$c_1c_2..c_n$]	c_1～c_n の中の任意の文字
[^$c_1c_2..c_n$]	c_1～c_n の中にない任意の文字
*r**	*r* に適合する文字列の 0 個以上の連続
¥.	ピリオド(¥で正規表現の特別な意味をなくす)

^The は行頭の The にマッチします。

[acgt]は a,c,g,t の任意の一文字にマッチします。

[acgt]*は a,c,g,t から構成される任意の長さの文字列(長さ０も含む)にマッチします。

[a-z]はアルファベットの小文字にマッチします。

課題：英単語”The”および”the”がある行を表示するスクリプトを書きましょう。

課題：英単語”The”が先頭にある行を表示するスクリプトを書きましょう。The の前にスペースがあっても許すものとします。

課題：アルファベット、空白以外の文字が含まれる行を表示するスクリプトを書きましょう。

6.フィールドセパレータ

$1,$2,$3 は処理している行の１番目の単語、２番目の単語、３番目の単語を表していました。より正確に言えば、入力行のスペースで区切られた部分が$1,$2,$3 の境目になっていました。Awk ではこの境目となるもの（フィールドセパレータ）を変えることができます。

次のテキストファイルを作って下さい。ファイル名を names.txt とします。

```
Tetsuya Iida @ center @ 2
Atsunori Inaba @ right @ 41
Atsuya Furuta @ catcher @ 27
```

そして次のような awk スクリプトをつくってみましょう。ファイル名を names.awk とします。

```
BEGIN { FS= “@ “}
{ print $2 }
```

そしてこのスクリプトを実行して下さい。次のような実行結果が得られるはずです。

```
center
right
catcher
```

この場合、”@”がフィールドセパレータになっています。従って center の列が第２フィールド($2)に相当するのです。

なお、BEGIN の行を書かなくても awk を呼ぶときに、

```
awk –F@ -f names.awk names.txt
```

とすれば同じように"@"がフィールドセパレータとなります。

課題：住所録を３人分ほど作ってみましょう。名前、住所、電話番号を","で区切って１行に１人分ずつ書き、awk を使って電話番号だけを３人分表示してみましょう。

7.awk 組み込み変数

awk にはいくつかもとから意味を持つ変数があります。今まで何気なく使ってきた$0,$1,$2などもそうです。NR は現在処理中の行が何行目かを表します。次のスクリプトを見てみましょう。

```
{ print NR":"$0 }
```

これはテキストファイルの中味を“行番号：”付きで表示します。実際に確かめてみましょう。

※print は画面上に変数の値、文字列などを出力する命令ですが、変数と文字列（ダブルクオテーション ”で囲む）を続けて書くことができます。上の例では変数 NR,文字列":"、変数$0 の中味が続けて表示されます。

変数 NR を使って偶数番目の行を出力することもできます。

```
NR % 2 = = 0 { print $0 }
```

NR % 2 = = 0 は NR を２で割った余りが０なら、つまり偶数なら、という条件文です。この条件に合った行が出力されることになります。

※このように{ 処理 }の左側には文字列による条件分だけでなく、数式による条件文を書くことができます。

課題：１行目から 50 行目までを行番号付きで表示するスクリプトを書いてみましょう。

この他に組み込み変数として NF(その行のフィールド数)などがあります。調べてみましょう。

8.awk 組み込み関数

awk にはいくつか便利な関数が用意されています。いくつかを紹介しておきましょう。

gsub(r,s,t)　　文字列 t の中で r に適合するもの全てを s で置換し、置換数を返す。

ファイル名：genome.awk

```
{ line = $0;
```

```
 nsub = gsub(“gene”, “genome”, line);
 print line;
 print nsub;
}
```

コマンド行

```
echo “Analysis of the genes” | awk –f genome.awk
1
```

※このようにechoなどのコマンドの出力結果をパイプ｜でつないでawkへの入力とすることができます。

実行結果

```
Analysis of the genomes
```

length(t)　文字列 t の長さを返す。

printf　書式に従って文字列や数値を表示する。C 言語のものとよく似ているので詳しくは各自調べて下さい。

split(s,a,fs)　fs をフィールドセパレータとして s を配列 a に分解し、できた配列数を返す。s は a[1],a[2],a[3],…に分解される。

ファイル名:skiing.awk

```
{ line = $0;
 split(line, array, “-“);
 print array[3];
}
```

コマンド行

```
echo “I-LIKE-SKIING-IN-WINTER” | awk –f skiing.awk
```

実行結果

```
SKIING
```

substr(s,p,n)　s の p 番目から始まる長さ n の部分文字列を返す。

ファイル名:mothers.awk

```
{ print substr($0, 2, 5) }
```

コマンド行

```
echo “Mothers” | awk –f mothers.awk
```

実行結果

```
other
```

課題：ある英文テキストファイルの中にアルファベット a,c,g,t がいくつずつ含まれているかを表示する awk スクリプトを gsub を使って書きましょう。

9.配列

substr 関数のところでちょっとでてきましたが、awk では配列変数を使うことができます。配列変数は変数名に数字をつけた変数で、似たような性質のデータを一元的に管理できるようになります。

使い方は **変数名[整数]** で、例えば、line[2] = “number 2”とすると line[2]に”number 2”という文字列が代入されます（C 言語のように配列の宣言をする必要はありません)。

[ ]の中の整数は変数にすることもできます。例えば i = 3; line[i] = “number three”とすれば line[3]の中に”number three”という文字列が代入されます。

10.if 文

if 文はある条件にあったときに指定した処理を実行します。書式は以下の通りです。

if (条件) 条件に合ったときの処理
else 条件に合わなかったときの処理省略可

偶数番目の行を表示するスクリプトは、

```
NR % 2 = = 0 { print $0 }
```

でしたが、

```
{ if (NR % 2 = = 0)print $0 }
```

と書いても同じように処理されます。

また、

```
/the/ { print $0 }
```

は

```
{ if ($0 ~ /the/) print $0 }
```

と同じです。

課題：連続して重複した行を１つにまとめて出力するスクリプトを書きましょう。

I like driving. I like watching baseball. I like watching baseball. I like watching baseball. I like playing soccer. I like playing soccer. I like watching baseball.	⟶	I like driving. I like watching baseball. I like playing soccer. I like watching baseball.

ちょっと難しいと思われる方のためにヒントです。たった 2 行で書けるはずです。

```
NR = = 1 { prev = $0 ;..??..??.?.?}
NR > 1{ if (prev != ???…??.???…? }
```

11 for 文

C 言語などでお馴染みの for 文が awk でも使えます。for 文はある条件が満たされている間、指定された処理を実行する命令です。書式は以下の通りです。

for(初期設定;処理継続条件;更新処理){
　　処理の内容
}

・for 文に入ったときに最初に**初期設定**が実行されます。
・**処理継続条件**に合う間、**処理の内容**が実行されます。
・**処理の内容**を終えるたびに**更新処理**が実行されます。

次の awk スクリプトは各行のフィールドを”,”で分けるように加工して出力します。
ファイル名:sepat.awk

```
{ for(i = 1;i < NF;i ++){  printf(“%s,”, $i); }
 print $NF;
}
```

入力ファイル名:

```
Bluebird   205PS   Nissan
Galant     205PS   Mitsubishi
Prelude    220PS   Honda
```

```
Supra     280PS   Toyota
```

実行結果:

```
Bluebird,205PS, Nissan
Galant ,205PS, Mitsubishi
Prelude ,220PS,Honda
Supra,280PS,Toyota
```

for(i = 1;i < NF;i ++)で i の値を 1 から NF(一行のフィールド数)の直前まで(NF-1 まで)
1 ずつ(i++)増やします。printf(“%s,”, $i)で i 番目のフィールドが”,”とともに出力されます。

応用課題：ある文字列を見つけたらその行と、その後５行を表示するスクリプトを作って下さい。

解法：ある行を見つけて、行を（５行分）表示中であることを示す変数（フラッグ）を使うと便利です。

```
BEGIN{ to_print = 0; }
/ 検索文字列 / { to_print = 5; }
to_print > 0 { ??????????? }
```

応用課題：ある文字列を見つけたらその行と、その前５行を表示するスクリプトを作って下さい。

解法：配列を使って行を記憶していくと便利です。

```
{ for(I = 5;I > 0;I --)prevline[I] = prevline[I – 1];
 prevline[0] = $0; }
```

応用課題：入力ファイルが次のような構成になっていると仮定します。

```
For sale                    ┐
        Product = car       │
                Premera ‘90 │  レコード１
        Price = ¥1,000,000  ┘
For rent                    ┐
        Product = car       │
                Legnum GDI  │  レコード２
        Price = ¥10,000 / day┘
For sale                          :
        Product = motorcycle      :
                Zephyr 400        :
        Price = ¥400,000
For sale
        Product = car
                Maclauren F1
        Price =  ¥90,000,000
```

(1) For sale のレコード全体を全て出力するスクリプトを書きましょう。

(2) Product が car のレコードを全て出力するスクリプトを書きましょう。