データベース入門 資料 3
テキストファイルのフィールド操作 (awk とパイプ)

平成 23 年 5 月 1 日

目 次

1 awk を用いた特定フィールド（列）の抽出

この章では，改行記号で区切られたレコード（行）から，特定の記号で区切られたフィールド（列）を抜き出す方法を学ぶ。

1.1 awk について

awk とその使用方法全般に興味があれば，次の文章で始まる GNU awk のユーザーガイド (Effective AWK Programming) を読んでみよう。

> awk の名前はその開発者達の名前 Alfred V. Aho, Peter J. Weinberger そして Brian W. Kernighan から来ているものである。awk のオリジナルバージョンは 1977 年に AT&T のベル研究所で開発された。
>
> 1985 年にはユーザー定義関数、複数の入力ストリーム、動的正規表現の導入によってより強力なプログラミング言語となったバージョンが開発された。この新しいバージョンは一般的には UNIX System V Release 3.1 で使えるようになった。System V Release 4 での新 awk では幾つかの新しい機能追加と、言語の”暗い隅”をきれいにする作業が行われた。POSIX のコマンド言語およびユーティリティの標準は gawk のデザイナーとオリジナルを開発したベル研究所の awk 開発者の両方からのフィードバックが反映されている。

インターネット上の日本語訳としては，Edition 1.0.4 版[1]がある。なお，この授業で利用する awk は GNU 版の gawk である。

1.2 レコードとフィールドについて

この資料でも前回の教材テキスト[2]を利用する。このテキストファイルの内容を，less コマンドを使って確認しておこう。コマンド行でファイル名をタイプするときには，Tab キーを使ったファイル名補完を使おう。

この教材テキストには，改行コードで区切られたがあり，その行の中に + (プラス) 記号で区切られた項目 (列) が存在する。awk では，各行を**レコード**と呼ぶ[3]。さらにその中の，ある記号で区切られた項目 (列) を**フィールド**と呼ぶ。

awk は，このような項目（列），すなわちフィールドを指定して，さまざまな操作をすることに適している言語である。

例えば，教材テキスト

> 単語表記+カタカナ読み+活用見だし語+品詞コード+ローマ字読み（改行コード）

のうち，1 列目の単語フィールドのみを取り出して less で閲覧するには，

```
gawk -F+ '{print $1}' /pub/db/data/75.60k.vocab.romaji | less
```

を実行すればよい。

[1] http://www.kt.rim.or.jp/~kbk/gawk-30/gawk_toc.html

[2] http://echoes.hak.hokkyodai.ac.jp/db/550/less_grep.html

[3] 「レコード」が改行コードで区切られている必然性はないが、以下では、とりあえず「改行」で区切られるものを「行」＝レコードとする。

1.3 awk を用いたフィールド操作の基本

1.3.1 実行書式

gawk の基本的な実行の書式は次のとおりである。

```
gawk [オプション] 'awk のプログラム命令' [対象とするファイル]
```

この授業では，gawk の各種オプションおよびプログラム命令のうち

1. レコード内フィールドの区切り記号を指定する -F オプション
2. 抽出したいフィールド番号を指定して出力する print 命令
3. そのフィールド番号記述方法（$番号）

の利用法の習得は必須である。

1.3.2 出力するフィールドを変更する

第 1.2 節で紹介した最初の awk の実行例

```
gawk -F+ '{print $1}' /pub/db/data/75.60k.vocab.romaji | less
```

において，$1 は区切り記号+で区切られた第 1 番目のフィールドを表している。では，$1 を $2 や $3 等に変更するとどうなるか，試してみよう。

さらに，$番号をカンマで区切って並べれば，フィールドの出力をいろいろと変更することができる。

```
gawk -F+ '{print $2, $1}' /pub/db/data/75.60k.vocab.romaji | less
```

次の例も試してみよう。

```
gawk -F+ '{print $0}' /pub/db/data/75.60k.vocab.romaji | less
```

実行結果からわかるとおり，$0 は処理中の行全体を格納する変数である。

注意 「$番号」の間にカンマを入れると awk は列を空白で区切って出力するが，カンマを入れずに

```
gawk -F+ '{print $2 $1}' /pub/db/data/75.60k.vocab.romaji | less
```

とすれば，$2 と $1 の間は区切られない（$2 と $1 を結合して出力する）[4]。

1.3.3 入力フィールドの区切り文字指定を変更する

教材テキストでは，列（フィールド）が + で区切られているが，/ を区切り文字とみなすように awk に指示してみよう。

```
gawk -F/ '{print $2}' /pub/db/data/ame | less
```

ここで，/pub/db/data/ame は，教材テキストから ame を含む行のみを取り出したファイルである。なお，これと同じ中身のファイルは前回の練習問題でも作成したので，代わりにそちらを使ってもよい。

[4] 空白以外の文字を出力フィールドの区切り文字にしたければ，'{OFS = "+"; print $2, $1}' のように awk の組み込み変数 OFS に区切り文字を代入するか，'{print $2 "+" $1}' とすればよい。

注意　フィールドの区切り文字指定 -F は awk のオプションなので，この指定は省略することもできる。そのとき awk は（連続する）空白文字やタブ文字を列の区切りとみなす。

2　パイプの利用

パイプ (|) とは，あるコマンドの出力を別のコマンドの入力に与えるものであり，パイプを使えば，あるコマンドの処理結果を，引き続き別のコマンドで処理することが可能になる。例えば，コマンドの出力を less で読む

```
コマンド | less
```

は，よく使われるパイプの一利用法である。

この章では，コマンドの標準入力とパイプについての要点を確認した上で，これまでに紹介した grep や awk 等をパイプと共に使うことにより，レコード（行）やフィールド（列）に対するより複雑な処理を行う方法を学ぶ。

なお，この章で紹介する標準入力やパイプについての詳細は，例えば,「情報機器の操作」テキスト[5] の「11. 標準入力の利用 — リダイレクトとパイプ」に説明がある。

2.1　ファイル名を与えずに awk を実行する—標準入力の利用

コマンド行にファイル名を指定せずに,

```
gawk '{print $1, $2}'
```

とだけ打って awk を実行してみよう。

新しいプロンプトが現れないのは，awk がまだ動いているためである。続いて，キーボードから

```
abc def ghi
```

と打ってエンターを押すと，`abc def` が出力される。さらに,

```
a bcd ef g
```

と打ってエンターを押すと，`a bcd` と出力される。

これらの出力が得られたのは，実行中の awk が `'{print $1, $2}'` の命令に従って，キーボードからの入力行を処理したためである。オプション -F を指定していないので，awk は空白を列の区切り文字とみなすことに注意しよう。

このように，ファイル名を指定しないで awk を実行すると，awk はキーボードからの入力（標準入力）を処理して出力する。この awk を正常終了させるには，ctrl-d (EOF; 入力の終わり) を押せばよい。

注意　多くの UNIX コマンドは，ファイル名を省いて実行すると，awk と同様に標準入力から入力を受け取る。

[5] `http://echoes.hak.hokkyodai.ac.jp/db/513/`

2.2 コマンド出力を別のコマンドの標準入力に与える—パイプ

先に実行した

```
gawk -F+ '{print $1}' /pub/db/data/75.60k.vocab.romaji | less
```

と同じことを，awk の引数にファイル名を与えずに行うには，どうしたらよいだろうか。なお，前節では，ファイルの代わりにキーボードから abc def ghi 等の入力を与えたが，今回は，教材テキスト 75.60k.vocab.romaji の中身を全部キーボードから打ち込む訳にはいかない。

答えは，cat とパイプを使って，

```
cat /pub/db/data/75.60k.vocab.romaji | gawk -F+ '{print $1}' | less
```

とすればよい。

これを実行したときの動作は次のとおりである。なお，ここでは「`| less`」については考えず，その左側だけに注目する。

- cat が教材テキストの中身を出力する。
- awk の引数にはファイル名が与えられていないので，awk は標準入力からデータを受け取る。
- パイプ | は cat と awk の仲立ちをし，cat の出力（教材テキストの中身）を awk の標準入力に直接流し込む。

  awk 単体での実行時にファイル名を略すと標準入力はキーボードになるが，この実行例ではパイプによって awk の標準入力が cat の (標準) 出力に結ばれた。

パイプを使うときには，ファイル名を指定すべきコマンドがどれなのかに注意すること。

2.2.1 例 1: grep と awk を組み合わせて特定行の特定列のみを抽出する

grep の実行例

```
grep asshukukuki /pub/db/data/75.60k.vocab.romaji
```

とパイプおよび gawk を組み合わせて，教材テキストの中の asshukukuki を含む行 (レコード) の，単語表記列 (フィールド) だけを出力する。

```
grep asshukukuki /pub/db/data/75.60k.vocab.romaji | gawk -F+ '{print $1}'
```

2.2.2 例 2: grep して awk して sort する

sort は文字通り，行を並べ替えて出力するコマンドである。ここでは，他のコマンドからの出力を，逆順に並べ替えてみる。

逆順オプション `-r`

次の例が何をするのか，よく考えてから実行してみよう。

```
grep ame /pub/db/data/75.60k.vocab.romaji | gawk -F+ '{print $5, $1}' | sort -r | less
```

2.3 演習問題

1. 教材テキストの中の ame を含む行のみから第 1 フィールドのみを出力し，less で閲覧しなさい。grep と awk をつかうこと。
2. 教材テキストから，yuki を含む行（レコード）の中の「ローマ字読み」の列（第 5 フィールド）のみを抽出し，それをアルファベット順に並べ替えて出力しなさい grep, awk, sort をつかうこと。
3. 教材テキストのうち，ame を含み、かつ、44 を含まない行のみを取り出して less で閲覧しなさい。パイプと grep のオプションを使うこと。
4. 教材テキストから，`ken'i` を含む行（レコード）の中の「ローマ字読み」の列（フィールド）のみを抽出しなさい。(ヒント：特殊文字の意味を無効化)
5. ホームディレクトリで ls -al を実行しなさい。その結果から，4 を含む行のみを表示しなさい。パイプを使うこと。
6. `ls -l` の出力のうち,「ファイル名」(9 列目) と「ファイルサイズ」(5 列目) の列のみを出力しなさい。パイプを使うこと。

3 発展 – awk における pattern 指定処理

“Effective AWK Programming” には

> awk の基本的な機能は、ファイルからあるパターンを含んでいる行 (もしくは他のテキストの構成単位) を検索することである。ある行がパターンの一つにマッチしたとき、awk は特定のアクションをその行に対して実行する。 awk はこのようにして入力ファイルの最後の行までそれぞれの行を処理し続ける。

と記述されている。ここにあるとおり，awk への「プログラム命令」を

pattern {*action*}

の形式で与えれば[6]，awk は指定したパターン (*pattern*) の行に対してのみ動作 (*action*) を行う。すなわち，ある文字列を含む行のみを awk で処理したければ，*pattern* として,「/文字列/」 を指定すればよい。

例えば,

```
gawk -F+ '/ame/ {print $1}' /pub/db/data/75.60k.vocab.romaji
```

は，入力行に ame を含む行の第 1 フィールドのみを出力する。これは

```
grep ame /pub/db/data/75.60k.vocab.romaji | gawk -F+ '{print $1}'
```

を実行するのと同じである。

また,「/文字列/」に代え,「/正規表現/」の指定も可能である。正規表現については，前回のテキストを参照のこと。

次の例がどのような結果を出力するかを考えて，実行してみよう。

[6] この形式を複数並べて与えることも可能

```
gawk -F+ '/ame/ {print $0}' /pub/db/data/75.60k.vocab.romaji

gawk -F+ '/n..kuni/ {print $0}' /pub/db/data/75.60k.vocab.romaji
```

3.1 練習

1. s を含み行末が kuki である行の「単語表記」フィールド (1 列目) のみを出力しなさい。grep と awk をつかうこと。
2. 前章の 2.3 演習問題のうち，設問 1 と 5 を，grep を使わずに解きなさい。