情報科学演習／データベース入門 A 資料 4
テキストファイルの閲覧と文字列の検索

平成 29 年 5 月 1 日

目 次

1 less を使ったファイル閲覧と文字列検索

1.1 ページャーと less

テキストファイルの中身を閲覧する道具として cat コマンドやエディタがある。その他にもページャー (pager) という種類のコマンドがあり，ページャーを使えば，比較的大きなテキストファイルの中身を手軽に閲覧することができる。more コマンドは UNIX に標準的に備わっている代表的なページャーであるが，ここでは more の機能拡張版である less を使ってファイルを閲覧する。さらにテキストファイル内から必要な情報を取得するために検索操作を行ってみる。

なお，コマンドの使いかたを調べる際に使う man コマンドは，マニュアルの表示に more か less を使うことが多い。また，この授業の後半で使うデータベースソフトウェアにおいても，検索結果の表示にページャーを使う。このため，この授業における今後の実習のためにも more や less の基本的な使い方は習得する必要がある。

1.2 教材テキスト

ここで扱うファイルの名前は 75.60k.vocab.romaji であり，新聞記事から頻出単語約 60,000 語を抽出したテキストファイルである。このファイルは次のディレクトリ内ある。

ディレクトリの絶対パス名 /pub/eis/data

75.60k.vocab.romaji には，各行につき単語の

- 表記
- カタカナ表記の読み
- 活用語の見出し語
- 品詞タグ (品詞を数値でコード化したもの)
- ローマ字表記の読み

が記述されていて，各項目は + の記号で区切られている。

1.3 less の起動法

less の基本的な起動法は次のとおりである。

```
less filename
```

ここで，*filename* には内容を見たいファイルの名前（ファイル名）を指定する。

ただし，通常，コマンドにはファイル名に代えてファイルのパス名を与えても実行できるため，ファイル名という用語をパス名も含む広義の意味で使うことがある。less に与える引数にも，ファイル名に代えてファイルのパス名を与えてよい。

1.4 less の主要コマンド

下記の各コマンドは，less でテキスト文書を閲覧している最中に使える，less が持っているコマンドである。

機能	コマンド	コメント
終了	**q** (または **:q**)	
ヘルプ	**h**	
ページ移動	<SPACE> (または CTRL-f または **f**)	次ページに移動する
	b (または CTRL-b)	前ページに移動する
	j (または <ENTER> または ↓)	1 行下に移動する
	k (または ↑)	1 行上に移動する
	G	最後の行に移動する
	*n***G**	第 *n* 行に移動する
テキストを検索する	/*pattern* <ENTER>	*pattern* が最初に現れる位置に移動する (下方検索)
	?*pattern* <ENTER>	*pattern* が最初に現れる位置に移動する (上方検索)
	n	一番最近行った検索を繰り返す
	N	一番最近行った検索を逆方向に繰り返す

1.5 練習

1. まず，ファイル 75.60k.vocab.romaji の内容を表示するために cat コマンドを実行してみよう。実行後、適当なところで CTRL-c を押して，cat コマンドを終了させること。

2. less コマンドでファイルの中身を見てみよう。less では，<SPACE> を押して，次のページを閲覧できる。その他の操作は第 1.4 節の less の主要コマンドを参照のこと。

3. `kaiba` を検索語として検索することによって，カイバという単語を漢字ではどう記すのかを調べなさい。less で検索するには / に続いて検索語 (pattern) を記入し <ENTER> を押す。検索語の記入途中でタイプミスをしたら，CTRL-c で検索語入力を中止できる。

4. 新聞記事に現れた「教育」を含む単語にはどのようなものがあるか，検索を繰り返して調べなさい (n コマンドを使う)。検索語は `kyoiku` である。

5. 時間があれば，`man less` で less のコマンドを調べたり，less の操作をいろいろと試してみよう。

2　grep を用いた特定行（レコード）の抽出

この章では，grep コマンドを用いて，テキストファイルから特定の文字列を含む行のみを取り出して，閲覧する方法を学ぶ。

2.1　grep の基本的な使い方

grep の基本的な実行の形式は次のとおりである。

grep 文字列 ファイル名

この形式では，grep はファイルの内容から，特定の文字列を含む行のみを出力する。

2.1.1　実行例

以下の実行例において，ファイル名の指定には，tcsh シェルの入力補完機能（tab キーを押す）を使うのがよい。

まず，asshukukuki （圧縮空気）を含む行だけをファイルから取り出してみる。

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

次に asshuku （圧縮）を grep する。

```
grep asshuku /pub/eis/data/75.60k.vocab.romaji
```

「圧縮 (asshuku)」は欲しいけれども「合宿 (gasshuku)」はいらないならば，教材テキストでの項目（列）区切りが + （プラス記号）であることを利用して，

```
grep +asshuku /pub/eis/data/75.60k.vocab.romaji
```

を実行すればよい。

2.2　grep の出力を less で読む

「あめ」を調べたいと思って

```
grep ame /pub/eis/data/75.60k.vocab.romaji
```

とすると，407 単語（407 行）が該当し，結果を画面に表示しきれない。grep の出力を，パイプ (|) を使って less に入力すると，結果を less で 1 ページずつ見ることができる。

```
grep ame /pub/eis/data/75.60k.vocab.romaji | less
```

less の終了方法などについては，第 1.4 節の less の主要コマンドを参照のこと。

2.3 grep の一般書式

grep [options] *pattern* [*file...*]

grep は *file* で名前を指定された入力ファイル (*file* が指定されてないか， *file* の部分に - が指定された場合は標準入力) を読み込み，与えられた *pattern* にマッチする部分を含む行を探す。

以下にしばしば使う grep のオプションを挙げるので試してみよう。さらに詳しく知りたいときには `man grep` を実行すればよい。

2.3.1 よく使う grep のオプション

`-n`：各出力行の前に，入力ファイルにおける行番号を表示する
`-r`：ディレクトリ下のすべてのファイルを再帰的に読み取る
`-v`：マッチした行を表示しない。(マッチしない行を表示)
`--help`：簡単なヘルプメッセージを出力する
`--version`：grep コマンドのバージョンを出力する

上記のうちで，`--help` と `--version` は多くのコマンドで共通に使えるオプションであり，これらでは引数の *pattern* は不要である。

2.4 練習問題

1. 教材テキスト (75.60k.vocab.romaji) から yuki を含む行のみを表示しなさい。
2. grep のオプションを使って，教材テキストに yuki を含む行が何行あるかを表示しなさい。必要なオプションは man コマンドで調べること。
3. 教材テキストのうち，ame を含む行のみを，リダイレクト (>) を使ってファイルに格納しなさい。ただし，格納先のファイルは，ホームディレクトリに存在するディレクトリ eis17 の下の ame とする。eis17 が存在しなければ，まず作成すること。
4. ファイル ame の中から 44 を含まない行のみを抽出し，パイプと less で閲覧しなさい。

3 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。

3.1 awk について

awk とその使用方法全般に興味があれば，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。

インターネット上の日本語訳としては，Edition 3.1.6 版がある。なお，この授業で利用する awk は GNU 版の gawk である。

3.2 レコードとフィールドについて

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[1]。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 列目の単語表記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/eis/data/75.60k.vocab.romaji | less
```

を実行すればよい。

3.3 awk を用いたフィールド操作の基本

3.3.1 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。

3.3.2 出力するフィールドを変更する

第 3.2 節で紹介した最初の awk の実行例

```
gawk -F+ '{print $1}' /pub/eis/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。

[1]「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。

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

次の例も試してみよう。

```
gawk -F+ '{print $0}' /pub/eis/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/eis/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）[2]。

3.3.3　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/eis/data/ame | less
```

ここで，/pub/eis/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。これと同じ中身のファイルは前の練習問題でも eis17 に作成したので，代わりにそちらを使ってもよい。

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。

3.4　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すぐに現れないのは，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してみよう。`abc def` と出力される。さらに，

```
a bcd ef g
```

[2]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。

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 は標準入力 (通常はキーボード) からの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。

3.5　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_a/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。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，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_a/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。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。

3.5.1　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_a/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_a/data/75.60k.vocab.romaji | gawk -F+ '{print $1}'
```

3.5.2 例 2: grep して awk して sort する

sort は文字通り，行を辞書順に並べ替えて出力するコマンドである。sort コマンドに *file* を読み込ませて，辞書順に行を並べ替えて出力するには

`sort` *file*

とすればよい。*file* を省略すると sort も標準入力からデータを読み込む。ここでは，他のコマンドからの出力を sort コマンドに与え，辞書の逆順に並べ替えてみる。

さて，次の例が何をするのか，よく考えてから実行してみよう。sort コマンドのオプション `-r` は逆順に並べ替えるものである。

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

3.6 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. 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。
5. `ls -l` の出力のうち,「ファイル名」(10 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。

4 awk における pattern 指定処理

AWK のユーザーガイドには

> 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[3] で与えれば，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。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。

次の例が何をするのか考えてみよう。

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

4.1 練習

前章の練習問題 1 や 2 を grep を使わずに解いてみよう。

[3] この形式を複数並べて与えることも可能