フィルタコマンドの使い方

フィルタとは？

一般的にはフィルタとは，与えられたものの特定成分を取り除いたり，弱めたりする機能を持つものをいう（コーヒーのフィルタ，レンズのフィルタ，電気回路のフィルタ，ディジタルフィルタなど）．
Unix では，入力されたデータを加工して出力するプログラム（コマンド）をフィルタと呼ぶ．ここでは，Unix の代表的なフィルタコマンドとして次のものを取り上げる．

コマンド	機能
grep	ファイルの中から指定された文字列を探す
sort	ファイルから読んだ行を並べ換える
head	ファイルの先頭から指定された数の行を出力する
tail	ファイルの末尾から指定された数の行を出力する

ファイルの準備

フィルタを使う前に，まず cat コマンドを使ってファイルを作っておこう．ファイル名は testdata とする．（注：以下において▼はエンターキーを押すことを表すが，▼は表示されない．）

```
cat > testdata▼
```

このように入力したら，続いてデータを入力する．入力を間違えると実行結果が違ってくるので，間違えないようにしよう（もし間違えたら，最初から入力し直すこと）．ちなみに，入力した文字列は代表的なプログラミング言語の名前である．データが入力できたら Ctrl+D（コントロールキーを押しながらＤを押す）で，cat コマンドから抜け出るようにすること．

```
$ cat > testdata▼
BASIC▼
LISP▼
Pascal▼
C▼
ALGOL▼
APL▼
FORTH▼
COBOL▼
Ada▼
FORTRAN▼［エンターキーの後で Ctrl+D］
$ ■
```

この操作により testdata というファイルが作成される．cat は，引数（ひきすう）としてファイルを指定しないと，標準入力であるキーボードから入力した文字列を出力するが，その出力先がリダイレクションにより testdata というファイルになるのである．
次のようにして，正しく入力されているか確認してみよう．

```
cat testdata▼
```

grep コマンド

ファイルが準備できたら，実際に grep コマンドを使ってみよう．grep コマンドは，指定された文字列の検索を行うフィルタであり，次のような書式で使用する．

grep 文字列 ファイル名▼

Unix のコマンドでは，文字列は「'」（シングルクォーテーション）で囲むのが一般的だが，文字列の間に空白や特殊記号を含まない場合などには省略することができる．以下では省略することにする．
では，実際に次のように入力してみよう．

grep FORT testdata▼

このようにコマンドを入力すると，testdata の中から「FORT」という文字列を含む行を検索して画面に表示する．

```
$ grep FORT testdata▼
FORTH
FORTRAN
$ ■
```

なお，「-i」オプションを指定しない限り，文字列の小文字と大文字は区別される．次の２つの場合，どのような出力となるか予想してから，実際に試してみよう（「-i」オプションも試してみよう）．

grep a testdata▼

grep A testdata▼

sort コマンド

よく利用されるフィルタとして並べ換え（ソート）を行う sort コマンドがある．次のような書式で使用する．

sort ファイル名▼

この sort コマンドを使ってみよう．次のように入力すると，ファイルから読んだ行を，辞書の項目と同じ順に並べ換えたものを出力する．

sort testdata▼

```
$ sort testdata▼
ALGOL
APL
Ada
BASIC
C
COBOL
FORTH
FORTRAN
LISP
Pascal
$ ■
```

※注意※
sort は行を辞書式順序（辞書の項目と同じ順序）に並べ換える．各文字の順番は Unix の内部表現（整数値）によって決まる．上の例で，APL が Ada より先なのは，大文字（A-Z）の方が小文字（a-z）よりも小さい内部表現を持っているからである．

今度は「-r」オプションを指定して使ってみよう．次のようにすると，sort コマンドは逆順に並べ換えて出力する．

sort -r testdata▼

```
$ sort -r testdata▼
Pascal
LISP
FORTRAN
FORTH
COBOL
C
BASIC
Ada
APL
ALGOL
$ ■
```

head コマンド

head コマンドは，ファイルの最初の部分を表示する．

head -n 数字 ファイル名▼

とすると，指定された数の行だけファイルの先頭の行が表示される．

head -n 2 testdata▼

```
$ head -n 2 testdata▼
BASIC
LISP
$ ■
```

もし「-n 数字」を省略して，

head ファイル名▼

とすると，適当な数の行だけ出力される（行数はシステムの設定による）．また，ファイル名を複数個書くと，それぞれのファイルの先頭の行が指定された行数だけ表示される．

tail コマンド

tail コマンドは，ファイルの最後の部分を表示する．

tail -n 数字 ファイル名▼

とすると，指定された数の行だけファイルの末尾の行が表示される．

tail -n 2 testdata▼

```
$ tail -n 2 testdata▼
Ada
FORTRAN
$ ■
```

tail は，最後の行を表示すること以外は，head とほとんど同じである．「- n 数字」を省略した場合も同様である．

標準入力から読む

フィルタは，引数として指定されたファイルからデータを読み込む替わりに，標準入力からデータを読むことができるように作られている．これまでに説明したフィルタも例外ではない．

つまり，ファイルと同じ内容を標準入力（キーボード）から入力しても構わない．ただし，grep は，指定された文字列がある行を探し出したら，すぐにその行を出力する．次のように入力して確認してみよう．

```
$ grep FORT▼
LISP▼
FORTH▼
FORTH     ←この行は grep が出力した
BASIC▼［エンターキーの後で Ctrl+D］
$ ■
```

また，標準入力が使用できるということは，リダイレクト記号「<」を使って，ファイルの内容を grep に入力することもできるということである．

つまり，ファイルを引数とする方法

grep FORT testdata▼

の替わりに，ファイルの内容をリダイレクトで読み込む方法

grep FORT < testdata▼

としても同じ結果になる．

```
$ grep FORT < testdata▼
FORTH
FORTRAN
$
```

他のフィルタコマンドも同様で

コマンド ファイル名▼

という書式のものは

コマンド < ファイル名▼

としても同じ結果が得られる．sort，head，tail でも試してみよう．

実は，フィルタに限らず，Unix においてファイルを引数とするコマンドは，標準入力からデータを読めるように作られている．たとえば，cat testdata でも cat < testdata でも出力結果は同じである．

フィルタをパイプでつなげる

フィルタコマンドは標準入力から読み込むことができ，また，標準出力に書き出すことができるということは，フィルタはパイプでつなげて使うことができるということを意味する．

たとえば，

```
sort testdata | head -n 5▼
```

というコマンドは，sort testdata の標準出力が head -5 への標準入力となり，その結果が標準出力（画面）に出力（表示）される．どのような結果になるかは，実際に試してみよう．

もちろん，コマンドの順序が異なれば，一般的には結果も異なる．

```
head -n 5 testdata | sort▼
```

とすれば，どのような出力となるか予想してから，実際に試してみよう．

また，当然のことながら，複数のパイプを使うこともできる．

```
grep -i a testdata | sort | tail -n 3▼
```

とすれば，どのような出力となるか予想してから，実際に試してみよう．

その他のフィルタコマンド

Unix には，ここで紹介したフィルタ以外にも，tee，cut，uniq，wc などのような比較的シンプルなコマンドや sed や awk のような高機能なプログラムもある．それらについても，man ページやインターネットで使い方を調べてみよう．

※注記※

head と tail では「-数字」というオプションも可能である．たとえば，

```
head -2 testdata▼

tail -2 testdata▼
```

としても，先と同じ結果が得られる．このオプションは現在では推奨されていないとのことであるが，こうした使用例も見かけることがあるので覚えておくとよいだろう．