Unix コマンドの世界

――cat が分かればすべてが分かる！？――

小　堀　　聡
Satoshi KOBORI

理工学部電子情報学科　教授
Professor, Department of Electronics and Informatics

1. はじめに

この記事は，授業などで Unix の基本を一通り学習したけれど，今ひとつコマンドの使い方が分からないという学生諸君を対象にしている．ここでは，Unix コマンドのまとめ（復習）として，cat コマンドを例にして説明していく．cat は Unix の基本コマンドがもつ特徴，すなわち**オプションや引数**の与え方はもちろん，**標準入力，標準出力，リダイレクション，パイプ**などをすべて備えているので，「cat が分かれば Unix 基本コマンドのすべてが分かる」と言ってもいいぐらいである．

以下の例については，必ず Unix 環境にログインして実際に実行しながら，内容を確認していくことにしよう．ただし，もし何か分からないことに出くわしたら，もっと基本的なテキストをよく読み直したうえで操作を進めていくこと．

2. ファイルの準備と確認

この段階では一切の説明なしに，cat コマンドの学習に必要となるファイルを準備する．

まず，cat コマンドを使って 2 つのファイル（ファイル名は file 1 と file 2）を作っておこう．それぞれについて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 コマンドの標準入力状態から抜け出す．（注：これ以降，▼はエンターキーを押すことを表すが，▼は表示されない．また，■はコマンド行カーソルを示す．）

```
$ cat  >  file 1▼
aaa▼
This is file 1. ▼[Ctrl＋D]
$ cat  >  file 2▼
bbb▼
This is file 2. ▼[Ctrl＋D]
$ ■
```

次に，作成したファイルの内容を表示させて確認しておこう．

```
$ cat file 1▼
aaa
This is file 1.
$ cat file 2▼
bbb
This is file 2.
$ ■
```

3. ファイルの連結——cat 本来の機能

cat コマンドの本来の機能はファイルを連結することにあり，その名前は「結びつける」という意味の英単語 concatenate（あるいは catenate）に由来している．連結するとはどういうことか，次の例で確かめてみよう．

```
$ cat file1 file2▼
aaa
This is file1.
bbb
This is file2.
$ ■
```

ここで確かめられることを以下に挙げる．

(1) 1 行目のように，cat は指定された複数のファイルの内容を指定された順に連結し，標準出力に書き出す．

(2) cat という文字列のあとに示したファイル名が引数となる．上の例では file1 と file2 が引数である．

(3) 標準出力は通常，ディスプレイにつながっているので，**「標準出力に書き出す」＝「ディスプレイに表示」**となる．上の例では file1 と file2 の内容が（並べた順に）連結されて画面に表示されている．

なお，cat の引数で指定するファイルの数は 2 つとは限らず，3 つ以上でも構わない（後述するようにファイルが 1 つ，あるいはファイルがない場合もある）．たとえば，次のような表記も可能である．

```
cat file1 file2 file1 file2
```

また，ファイルの指定にはワイルドカードを用いることもできる．たとえば，次のような表記も可能である．

```
cat file?
cat file*
```

※これらの例ではそれぞれどのように表示されるか考えてみよう．ワイルドカードを用いる場合ではどのファイルがどの順序で連結されるかを考慮する必要がある（ディレクトリに存在するファイルによって結果が異なる）．

4. ファイルへの出力
——リダイレクション“>”を使う

cat コマンドで連結したファイルの内容をディスプレイにではなく，ファイルに出力することもできる．次の例で確かめてみよう．

```
$ cat file1 file2 > file3▼
$ cat file3▼
aaa
This is file1.
bbb
This is file2.
$ ■
```

これは，書き出す先を標準出力（ディスプレイ）からファイルに切り替えたにすぎない．上記の例では file1 と file2 の内容を連結し，file3 というファイルに出力している（もし file3 というファイルが存在しなければ新規に作成され，存在しているならば上書きされる）．

このように標準出力から他の出力先に切り替えることをリダイレクションといい（もしくは切り替えること自体をリダイレクトするという），出力のリダイレクション記号として「>」が用いられる．

5. ファイル内容の表示
——1 つだけファイル指定

cat の引数で指定するファイルの数が 1 つの場合，他に連結するファイルがないので，ファイル内容が標準出力にそのまま出力される．すなわち，

```
cat file1
```

によって，file1 の内容がディスプレイに表示されるのである．

また，ファイル内容の表示とリダイレクションを組み合わせることで，ファイルのコピーが行える．

次の例を確認してみよう．

```
$ cat file 1▼
aaa
This is file 1.
$ cat file 1 > file 4▼
$ cat file 4▼
aaa
This is file 1.
$ ■
```

これにより，file 1 の内容が file 4 にコピーされる．もちろん，ファイルのコピーには cp コマンドが用いられるので，通常このようなことを行うことはない．通常使う cp コマンドも内部で同じことをやっていると考えてよい．

6. ファイルの作成――ファイル指定なしの場合

cat は引数としてファイルが 1 つも指定されない場合は標準入力から読み込む．標準入力は通常，キーボードにつながっているので，**「標準入力から読み込む」＝「キーボードから入力」**となる．

リダイレクションを使えば，標準入力（キーボード）から入力した内容をファイルにすることができる．このことを次の例で確かめてみよう．

```
$ cat > file 5▼
abcde▼
This is file 5.  ▼[Ctrl＋D]
$ ■
```

cat の直後がファイル名ではなくリダイレクション記号「>」であり，引数としてファイルが 1 つも指定されていないことに注意しよう．リダイレクション記号の後の file 5 は出力先のファイルであり，標準出力（ディスプレイ）からファイルに切り替えたことを示している．すなわち，標準入力（キーボード）から文字列を入力し，Ctrl＋D で入力状態を終了し，出力先として file 5 に書き込み，ファイルが作成される．これが一番最初に 2 つのファイル（file 1 と file 2）を作成した際に用いた方法なのである．

7. ファイルからの入力
――リダイレクション“<”を使う

読み込み元を標準入力（キーボード）からファイルに切り替え，ファイルの内容を cat に入力することができる．このことを次の例で確かめてみよう．

```
$ cat < file 1▼
aaa
This is file 1.
$ ■
```

上の例では，file 1 の内容が読み込まれ，標準出力（ディスプレイ）に出力（表示）されている．これは，読み込む先を標準入力（キーボード）からファイルに切り替えたにすぎない．

このように標準入力から他の入力元（ほとんどがファイル）に切り替えることもまた，リダイレクションといい，入力のリダイレクション記号として「<」が用いられる．

なお，上の例は，cat コマンドにおいてファイルを引数として直接指定する場合，つまり，

```
cat file 1
```

と同じ出力結果となる．ただし，結果が同じになるのは指定ファイルが 1 つの場合のみである（その理由を実際に試して考えてみよう）．

また，リダイレクションにより標準入力と標準出力の両方をファイルに切り替えることで，ファイルのコピーが行える．次の例を確認してみよう．

```
$ cat < file 1 > file 6▼
$ cat file 6▼
aaa
This is file 1.
$ ■
```

この例では「<」により file 1 の内容が cat に

入力され，「>」によって file 6 に出力されている．その結果，file 1 の内容を file 6 にコピーしたことになる．

さて，cat コマンドの動作をまとめて解説すると次のようになる．

(1) cat の引数としてファイルが複数指定　⇒　指定ファイルを連結して出力

(2) 出力先を標準出力（ディスプレイ）からファイルに切り替える　⇒　リダイレクション「>」

(3) cat の引数としてファイルが 1 つだけ　⇒　ファイル内容の表示

(4) cat の引数としてファイルの指定なし　⇒　標準入力からの入力状態　⇒　Ctrl＋D で入力終了

(5) 入力元を標準入力からファイルに切り替える　⇒　リダイレクション「<」

8. フィルタコマンドの使用方法

一般的に，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．フィルタコマンドには，代表的なものとして，grep, head, tail などがあるが，実は cat コマンドもフィルタコマンドに分類される．

フィルタコマンドは引数にファイルが指定されなければ，デフォルト（既定値）として

入力＝標準入力，出力＝標準出力

という動作上のルールがある．たとえば，cat コマンドを何の引数なしに

```
cat▼
```

と入力してもエラーとはならず，デフォルトの動作をする（どのような動作になるか，各自で考えてみよう）．

そして，このルールを利用してフィルタコマンドはパイプを使ってつなぐことができる．パイプ（パイプラインともいう）とは，1 つのデータの流れに対して，複数の処理を逐次的に（リレーのように）つなげて行う仕組みである．Unix コマンドラインでのパイプ処理では，下に示すように，パイプ記号「|」（バーティカルバー）を使って複数のコマンドをつなげる．

コマンド 1 | コマンド 2 | コマンド 3 …

このように「|」の左側のコマンドの標準出力が，「|」右側のコマンドの標準入力に接続される．上の例ではコマンド 1 の標準出力＝コマンド 2 の標準入力，コマンド 2 の標準出力＝コマンド 3 の標準入力となる．「|」の数はいくつでもかまわない．次の例を見てみよう．

```
cat file 1 file 2 | sort -r | head -n 5
```

この例では file 1 と file 2 の内容が連結されて sort コマンド，head コマンドの順に流し込まれ，その結果が標準出力に出てくる．

パイプ処理を使うと，各コマンドの結果を保存する必要がなくなる．上の例を別々に行おうとすると，次のようになる．

```
cat file 1 file 2 > temp 1
sort -r temp 1 > temp 2
head -n 5 temp 2
rm temp 1 temp 2
```

このように temp 1, temp 2 などの一時ファイルを作成する必要がある．パイプを使った方が明らかに記述が簡潔であり，また実際の動作速度も速い．

9. Unix コマンドの一般的な使用方法

以上，cat コマンドを中心に説明してきたが，ここではそのことを踏まえて Unix コマンド一般について解説する．

Unix のコマンドはシェルが翻訳し，Unix に指示を伝えることができる文字列である．一般的にコマンドはたいてい引数つきで実行されるが，引数の種類よって次のような一般形になる．

コマンド名 [-オプション 1　オプション 1 引数 -オプション 2　オプション 2 引数…] [主引数 1　主引数 2　…　主引数 n]

※[　] の中は必要に応じて指定する．

※オプションや主引数はすべて 1 つ以上の半角空白またはタブで離されていなければならない．

引数は大まかに，コマンドの動作を条件づけるオプションを指定するものと，処理の対象として文字列やファイル名などを指定する，いわば「主引数」に分けることができる（※主引数という呼び名はここだけのもの）．通常，オプションの指定は「-」（マイナス記号）をつける．オプションにも引数を与えることがある．文字列は「'」（シングルクォーテーション）で囲むと，文字列であることを明示することができる．ファイル名はほとんどの場合クォートせずにそのまま与える．

コマンドによっては引数を必要としないもの（たとえば，フィルタコマンド）もあれば，逆に必ず処理対象を指定しなければならないコマンドもあるし，どちらでも動作するコマンドも多い．例をあげると，次のように分けられる．

(a) 引数が必要なもの（ないとエラーになる）
cp, mv, rm など

(b) 引数が不要なもの（あるとエラーになる）
pwd など

(c) 引数がなくてもよいもの（なければデフォルトの動作をする）
ls, cd など

オプションについても同様である．ls コマンドを例にとると，

ls（オプション・主引数なし ⇒ デフォルトの動作）

ls -l Dir1（オプション“l”（long format）を指定，対象として Dir1 を指定）

ls -lR Dir1 Dir2（オプション“l”と“R”（recursive）を指定，対象として Dir1 と Dir2 を指定）

など，柔軟な引数指定が可能である．

10. おわりに

これまで説明した cat コマンドは，Unix のコマンドの一例に過ぎない．実は cat コマンドの本来の役割はファイルの連結にあるのであり，ファイル内容を表示させたり，ファイルを作成したりするのは，cat コマンドの本来の機能ではなく，むしろ Unix のコマンドを実行しているシェルの機能（標準入出力やリダイレクション）によるものである．

Unix のフィルタコマンドなどにおいては，入力は（ファイルが引数に指定されていなければ）標準入力，出力は標準出力と定められているため，リダイレクションにより入力や出力を切り替えたり，パイプでつないだりすることができるのである．この原理は Unix 以外の他の様々な OS（オペレーティングシステム）にも取り入れられている．

Unix ではすでに多くのコマンドが用意されているが，自分で作成したプログラムも標準入出力をサポートするようにすれば，リダイレクションやパイプに対応させることができる．

Unix の考え方の基本には，それぞれの役割に特化したプログラムを組み合わせることで，複雑な機能を実現するという原理があり，リダイレクションやパイプが利用できることは，Unix の大きな魅力のひとつとなっている．