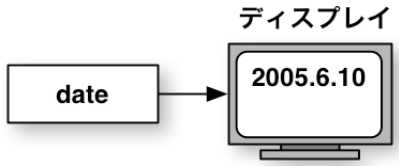
■ 標準入出力

□ コマンドの入力と出力

最も単純なコマンドのひとつ、date コマンドを例に、コマンドの出力について考えてみます。
まず date コマンドを実行してください。

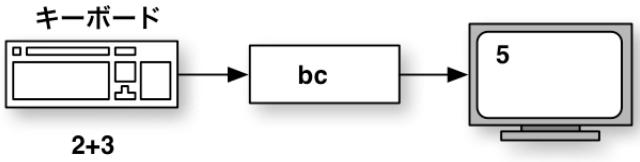
```
% date
2005 年  5 月 22 日 日曜日 04:05:49 JST
%
```



何らか日付や時刻を示す文字がディスプレイ上に表示されるでしょう。つまり date コマンドの実体はディスプレイ上に文字を出力するプログラムだったわけです。

次に bc コマンドを実行し、簡単な計算をさせてみます。以下の例では 2+3 と入力し、結果として 5 が表示されました。

```
% bc
2+3
5
quit
%
```



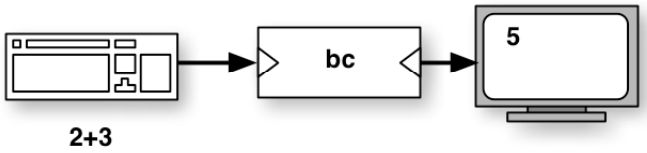
これは bc コマンドの実体が、文字をキーボードから入力し、結果としてディスプレイ上に文字を出力するプログラムだということを示しています。

□ 標準入出力

つまりこうしたコマンド（つまりその実体であるプログラム）には、右図のように何らか入力と出力があると考えられます(*)。

Unix プログラムには、こうした入力や出力が標準的に用意されています。これを標準入力、標準出力と呼びます。本稿では、説明を分かりやすくするために標準入出力を右図のような三角マーク（左右の△）で表現します。コマンドを示す長方形の左についた△が標準入力、右についた△が標準出力です。

通常、標準入出力はそれぞれキーボードとディスプレイに結びつけられています。
例えば bc コマンドは右図のように表現できます。

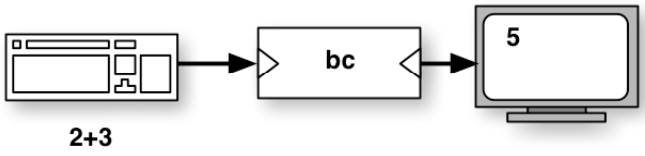


---

* 全てのコマンドがこうした入出力があるとは限りません。date コマンドには bc コマンドのような入力はありませんでしたし、xeyes のように GUI 上に効果をあらわすだけで、ターミナル上には何も出力しないコマンドも多くあります。

■ リダイレクション

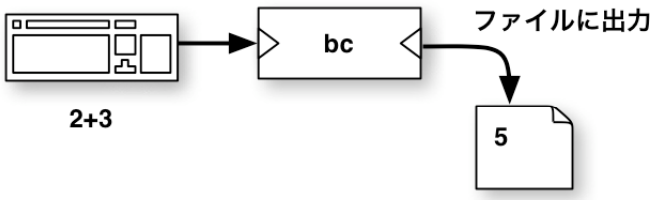
通常、標準入出力は右図のようにキーボードとディスプレイに結びつけられています。しかしこの結びつきは変更することができます。これをリダイレクションと呼んでいます。
（ re – direction ：向きを変える、程度の意味でしょうか。）



□ 標準出力の出力先を変更する

例えばコマンドを実行する際、以下のように「 > 」記号をコマンドの後に加えることで標準出力の出力先をディスプレイではなくファイルに変更することができます。

```
% bc > log.txt
2+3
quit
%
```



書式：
コマンド > ファイル名

機能：
・コマンドの出力をディスプレイではなく指定されたファイルにする。
・出力先のファイルが存在しなかった場合は新しく作成する。
・出力先のファイルが既に存在するものだった場合は元の内容を捨てて結果を上書きする。

例では bc コマンドの結果である「5」は log.txt ファイルの中に書き込まれます。右のようにして出力結果を確認すると良いでしょう。

```
% cat log.txt
5
%
```

□ 既存のファイルへの付け足し出力

「 > 」による標準出力のリダイレクションでは、出力先のファイルが既存だった場合に、そのファイルの中身を捨てて（消去して）、ファイルの先頭から結果を書き込みます。

以前の内容を捨ててファイルの先頭から書き込むのではなく、以前の内容を残してファイルの末尾に標準出力を書き足すこともできます。

右図のようにリダイレクションの指示に「 > 」ではなく「 >> 」（二つの > ）を用います。

```
% bc >> log.txt
10+20
quit
%
```

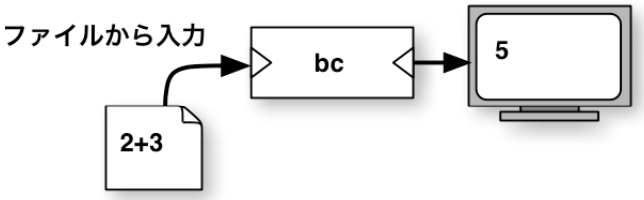
```
% cat log.txt
5
30
%
```

書式：
コマンド >> ファイル名

機能：
・出力先のファイルが存在しなかった場合は新しく作成する。
・出力先のファイルが既に存在するものだった場合は元の内容を捨てて結果を上書きする。

□ 標準入力の入力元を変更する

標準入力も同様にリダイレクション、つまり右図のように、標準入力の入力元をキーボードではなくファイルに変更することが可能です。
図では 2+3 という計算式がファイルに書き込まれており、それを bc の標準入力につないでいます。



実際の操作としては下図にあるように「 < 」記号をコマンドの後に加えます。下の例、左側は bc.in ファイルに計算指示が書き込まれていることを cat bc.in として確認し、右側では bc コマンドを実行する際に「 < bc.in 」と付け足して実行し、結果として 5 がディスプレイ上に表示されています。

```
% cat bc.in
2+3
%
```

```
% bc < bc.in
5
%
```

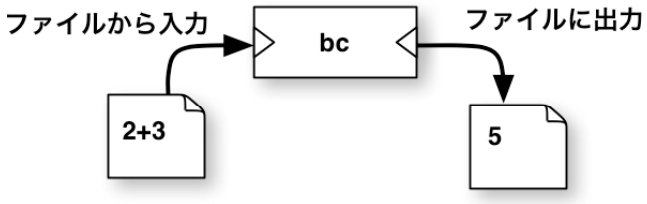
書式：
コマンド < ファイル名

機能：
・コマンドの入力をキーボードではなく指定されたファイルから得る。

□ 標準入力・標準出力を共に変更する

右図のように入出力の両方を変更することも良くあります。

この場合は単に「 < 」と「 > 」によるリダイレクションの指示を併記するだけです。
どちらを先に書くといった順序はありません。



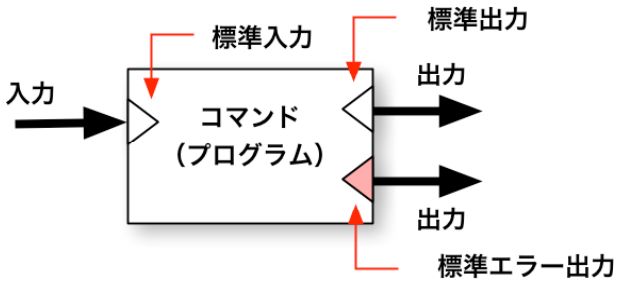
書式：
　コマンド < 入力元ファイル名 > 出力先ファイル名
または
　コマンド > 出力先ファイル名 < 入力元ファイル名

つまり下図のどちらでも同じように動作します。ともに bc.in ファイルから 2+3 という計算式を得て、log.txt ファイルに 5 という結果を出力するでしょう。

```
% bc < bc.in > log.txt
%
```

```
% bc > log.txt < bc.in
%
```

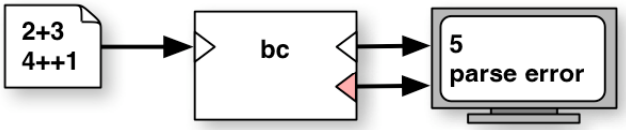
□ 標準エラー出力



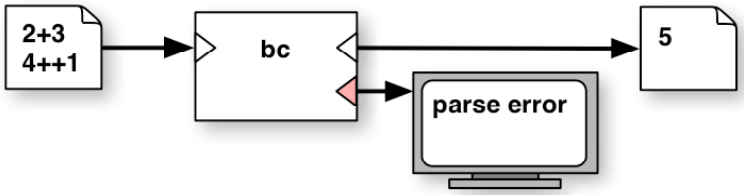
Unix プログラムにはもう一つ標準的に用意されている入出力があります。標準エラー出力と呼ばれるもので、エラーメッセージ等がここに出力されます。

通常、標準エラー出力は（標準出力と同様に）ディスプレイに結びつけられていますので、エラーメッセージなどはディスプレイ上にあらわれます。

例えば bc コマンドに計算式として不適当なもの（例では 4++1）を入力すればエラーメッセージが確認できます。



標準出力を「 > 」によってリダイレクションしても標準エラー出力はディスプレイに残ります。
そのため、エラーメッセージは指定されたファイルには書き込まれず、ディスプレイに表示されます。




```
% bc < bc.in > log.txt
(standard_in) 2: parse error
%
```

標準エラー出力をリダイレクションすることも可能ですが、ここでは説明しません。

□ 自作プログラムのリダイレクション

あなたの作ったプログラムについても標準入出力は備わっています。つまりあなたが作ったプログラムも、標準的にリダイレクションが可能です。
例えば「プログラミング A 」クラスで作成した「KSU KSU…」という文字を for( ) 文や while( ) 文を出力するプログラムも、以下に示すようにリダイレクションできます。

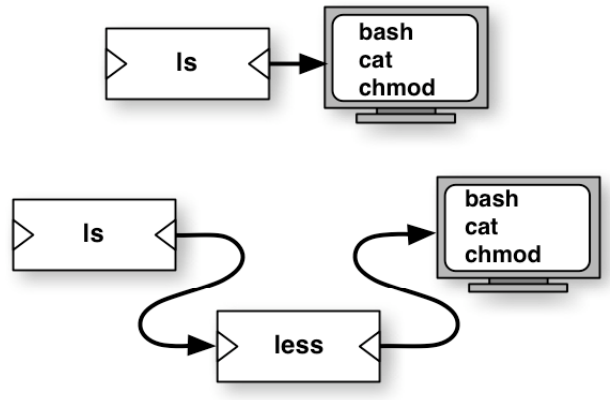
```
% cc for2.c
% ./a.out > a.txt
% cat a.txt
KSU KSU KSU KSU KSU KSU KSU KSU KSU KSU
KSU KSU KSU KSU KSU KSU KSU KSU KSU KSU
KSU KSU KSU KSU KSU KSU KSU KSU KSU KSU
KSU KSU KSU KSU KSU KSU KSU KSU KSU KSU
%
```

次節に示すパイプ機能も標準入出力の機構を利用したものですから、同様にあなたの自作プログラムにもパイプの機能を適用できます。

■ パイプ

標準入出力はファイルに向けるだけでなく、他の Unix コマンドに向けることもできます。つまりあるプログラムの標準出力を、あるプログラムの標準入力に接続することができるのです。

例えば ls /usr/bin などとすると大量の出力が画面上に表示されるでしょう。表示できる行数を超えると、そのまま以前の結果が流れていってしまいます。
これは不便なので一定行数ごとに表示を一旦停止させ、必要なら後戻りなどしながら結果を見たいわけですが、ls コマンドにはそうした機能はありません。ところが less コマンドはまさにそうした機能をもったコマンドであり、標準入力からの入力が可能です。
そこで以下のように「｜」記号を使って ls コマンドの出力を less コマンドの入力にむすびつけます。




```
% ls /usr/bin | less
```

書式：

コマンド｜コマンド…

機能：

・ 「｜」記号の左にあるコマンドの標準出力を、「｜」記号の右にあるコマンドの標準入力に結びつける。

注意：

・標準エラー出力はリダイレクション同様、パイプの影響を受けない。
・パイプは何段つないでも構わない。
・パイプとリダイレクションは組み合わせて使用可能だが、当然ながら「＜」は一番左のコマンドにしか書けないし、「＞」は一番最後のコマンドにしか書けない。

```
% bc < bc.in | sort | cat –n > log.txt
% cat bc.in
473088 + 223382
315056 + 223427
216605 + 337863
% cat log.txt
     1  538483
     2  554468
     3  696470
%
```

左の例は以下の処理を行います。

1. bc によって計算をし、
2. その結果を sort によって昇順（値の小さいものが前）に並び替え、
3. その結果に cat によって行番号をつけてファイルに出力する

（ほとんど意味のない例で申し訳ない）

図のように幾つものコマンドが数珠つなぎになって機能していることがわかるでしょう。